MPLS EM—MPLS LDP MIB—RFC 3815

The MPLS EM—MPLS LDP MIB - RFC 3815 feature document describes the MIBs that support the Multiprotocol Label Switching (MPLS) Label Distribution Protocol (LDP) based on RFC 3815, *Definitions of Managed Objects for the Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP)* , and describes the differences between RFC 3815 and the MPLS-LDP-MIB based on the Internet Engineering Task Force (IETF) draft Version 8 (draft-ieft-mpls-ldp-08.txt). RFC 3815 and IETF draft Version 8 provide an interface for managing LDP through the use of the Simple Network Management Protocol (SNMP).

In RFC 3815, the content of the MPLS-LDP-MIB is divided into four MIB modules: the MPLS-LDP-STD-MIB, the MPLS-LDP-ATM-STD-MIB, the MPLS-LDP-FRAME-RELAY-STD-MIB, and the MPLS-LDP-GENERIC-STD-MIB.

Cisco IOS MPLS Embedded Management (EM) is a set of standards and value-added services that facilitate the deployment, operation, administration, and management of MPLS-based networks in line with the fault, configuration, accounting, performance, and security (FCAPS) model.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for MPLS EM—MPLS LDP MIB - RFC 3815

- SNMP must be installed and enabled on the label switch routers (LSRs) or label edge routers (LERs).
- MPLS must be enabled on the LSRs or LERs.
- LDP must be enabled on the LSRs or LERs.
- Cisco Express Forwarding must be enabled on the LSRs or LERs.

For where to find configuration information for MPLS and LDP, see the Prerequisites for MPLS EM—MPLS LDP MIB - RFC 3815, on page 2.

Restrictions for MPLS EM—MPLS LDP MIB - RFC 3815

The implementation of the MPLS LDP MIB (RFC 3815) for Cisco IOS Release12.2(33)SRB is limited to read-only (RO) permission for MIB objects.

The following MPLS-LDP-STD-MIB tables are not implemented for Cisco IOS Release 12.2(33)SRB:

- mplsInSegmentLdpLspTable
- mplsOutSegmentLdpLspTable
- mplsFecTable
- mplsLdpLspFecTable
- mplsLdpSessionPeerAddrTable

The following MPLS-LDP-FRAME-RELAY-STD-MIB tables are not implemented for Cisco IOS Release 12.2(33)SRB:

- mplsLdpEntityFrameRelayTable
- mplsLdpEntityFrameRelayLRTable
- mplsLdpFrameRelaySessionTable

Information About MPLS EM—MPLS LDP MIB - RFC 3815

Label Distribution Protocol Overview

MPLS is a packet forwarding technology that uses a short, fixed-length value called a label in packets to determine the next hop for packet transport through an MPLS network by means of LSRs.

A fundamental MPLS principle is that LSRs in an MPLS network must agree on the definition of the labels being used for packet forwarding operations. Label agreement is achieved in an MPLS network by means of procedures defined in the Label Distribution Protocol (LDP).

LDP operations begin with a discovery (hello) process, during which an LDP entity (a local LSR) finds a cooperating LDP peer in the network and negotiates basic operating procedures between them. The recognition and identification of a peer by means of this discovery process results in a hello adjacency, which represents the context within which label binding information is exchanged between the local LSR and its LDP peer. An LDP function then creates an active LDP session between the two LSRs to effect the exchange of label binding information. The result of this process, when carried to completion with respect to all the LSRs in an MPLS network, is a label switched path (LSP), which constitutes an end-to-end packet transmission pathway between the communicating network devices.

LSRs use LDP to collect, distribute, and label binding information to other LSRs in an MPLS network, thereby enabling the hop-by-hop forwarding of packets in the network along normally routed paths.

MPLS EM—MPLS LDP MIB - RFC 3815 Feature Design and Use

RFC 3815 defines four MIB modules to support the configuration and monitoring of LDP. The MPLS-LDP-STD-MIB module defines objects that are common to all LDP implementations. To monitor LDP on an LSR or an LER, you need to use this MIB and one of the following Layer 2 MIB modules:

- MPLS-LDP-GENERIC-STD-MIB—Use this module and the MPLS-LDP-STD-MIB if the LSR or LER supports LDP that uses the global label space; for example, for Layer 2 Ethernet. This module defines Layer 2 per platform label space objects.
- MPLS-LDP-ATM-STD-MIB—Use this module and the MPLS-LDP-STD-MIB if the LSR or LER supports LDP that uses Layer 2 ATM. This module defines Layer 2 ATM objects.
- MPLS-LDP-FRAME-RELAY-STD-MIB—Use this module and the MPLS-LDP-STD-MIB if the LSR or LER supports LDP that uses Layer 2 Frame Relay. This module defines Layer 2 Frame Relay objects.

Note The MPLS-LDP-FRAME-RELAY-STD-MIB is not implemented for Cisco IOS Release 12.2(33)SRA.

If the LSR or LER uses LDP that supports Ethernet, ATM, and Frame Relay, then all four MIB modules need to be used by an SNMP agent on the LSR or LER.

The RFC 3815 upgrade to the MPLS-LDP-MIB is implemented to enable standard, SNMP-based network management of the label switching features in Cisco IOS software. Providing this capability requires SNMP agent code to execute on a designated network management station (NMS) in the network. The NMS serves as the medium for user interaction with the network management objects in the MIB.

The SNMP agent is a layered structure that is compatible with Cisco IOS software and presents a network administrative and management interface to the objects in the MPLS LDP MIB and, adds to the rich set of label switching capabilities supported by the Cisco IOS software.

You can use an SNMP agent to access MIB module objects using standard SNMP **get** and **getnext** commands to accomplish a variety of network management tasks. All the objects in the MPLS LDP MIB modules follow the conventions defined in RFC 3815, which defines network management objects in a structured and standardized manner.

Slight differences that exist between the RFC 3815 and the implementation of equivalent functions in the Cisco IOS software require some minor translations between the MPLS LDP MIB objects and the internal

data structures of Cisco IOS software. Such translations are accomplished by the SNMP agent, which runs in the background on the NMS workstation as a low priority process.

Cisco IOS Release 12(33)SRB supports the following MPLS LDP MIB-related functions:

- Generating and sending of event notification messages that signal changes in the status of LDP sessions
- Enabling and disabling of event notification messages by means of extensions to existing SNMP command-line interface (CLI) commands
- Specification of the name or the IP address of an NMS workstation in the operating environment to which Cisco IOS event notification messages are to be sent to serve network administrative and management purposes
- Storage of the configuration pertaining to an event notification message in NVRAM of the NMS

The structure of the MPLS LDP MIBs conforms to Abstract Syntax Notation One (ASN.1), thereby forming a highly structured and idealized database of network management objects.

MIB structure is represented by a tree hierarchy. Branches along the tree have short text strings and integers to identify them. Text strings describe object names, and integers allow computer software to encode compact representations of the names.

The MPLS LDP MIB is located on the branch of the Internet MIB hierarchy represented by the object identifier 1.3.6.1.2.1.10.166. This branch can also be represented by its object name iso.org.dod.internet.mgmt.mib-2.transmission.mplsStdMIB. The MPLS-LSR-STD-MIB is identified by the object name mplsLsrStdMIB, which is denoted by the number 4. Therefore, objects in the MPLS-LDP-STD-MIB can be identified in either of the following ways:

- The object identifier—1.3.6.1.2.1.10.166.4.[MIB-variable]
- The object name—iso.org.dod.internet.mgmt.mib-2.transmission.mplsStdMIB.mplsLdpStdMIB.[*MIB-variable*]

You can use any standard SNMP application to retrieve and display information from the MPLS LDP MIBs by means of standard SNMP GET operations. Similarly, you can traverse and display information in the MIB by means of SNMP GETNEXT operations.

Benefits of Using the MPLS EM—MPLS LDP MIB - RFC 3815 Feature

The MPLS LDP MIBs (RFC 3815) provide the following benefits:

- Retrieves MIB parameters relating to the operation of LDP entities, such as:
  - Well-known LDP discovery port
  - Maximum transmission unit (MTU)
  - Proposed keepalive timer interval
  - Loop detection
  - Session establishment thresholds
  - Range of Virtual Path Identifier (VPI)-Virtual Channel Identifier (VCI) pairs to be used in forming labels
- Gathers statistics relating to LDP operations, such as:

- Count of the total established sessions for an LDP entity
- Count of the total attempted sessions for an LDP entity

- Monitors the time remaining for hello adjacencies
- Monitors the characteristics and status of LDP peers, such as:
  - Internetwork layer address of LDP peers
  - Loop detection of LDP peers
  - Default MTU of the LDP peer
  - Number of seconds the LDP peer proposes as the value of the keepalive interval
- Monitors the characteristics and status of LDP sessions, such as:
  - Displaying the error counters
  - Determining the LDP version being used by the LDP session
  - Determining the keepalive hold time remaining for an LDP session
  - Determining the state of an LDP session (whether the session is active)
  - Determining the label ranges for platform-wide and interface-specific sessions
  - Determining the ATM parameters

MPLS LDP MIB (RFC 3815) Elements

The following functional elements of the MPLS LDP MIBs (RFC 3815) are used to perform LDP operations:

- LDP entity—Refers to an instance of LDP for purposes of exchanging label spaces; describes a potential session.
- LDP peer—Refers to a remote LDP entity (that is, a nonlocal LSR).
- LDP session—Refers to an active LDP process between a local LSR and a remote LDP peer.
- Hello adjacency—Refers to the result of an LDP discovery process that affirms the state of two LSRs in an MPLS network as being adjacent to each other (that is, as being LDP peers). When the neighbor is discovered, the neighbor becomes a hello adjacency. An LDP session can be established with the hello adjacency. After the session is established, label bindings can be exchanged between the LSRs.

These MPLS LDP MIB elements are briefly described in the following sections:

In effect, the MPLS LDP MIBs provide a network management database that supports real-time access to the various MIB objects within, describing the current state of MPLS LDP operations in the network. This network management information database is accessible by means of standard SNMP commands issued from an NMS in the MPLS LDP operating environment.

The MPLS LDP MIBs support the following network management and administrative activities:

- Retrieving MPLS LDP MIB parameters pertaining to LDP operations
- Monitoring the characteristics and the status of LDP peers

- Monitoring the status of LDP sessions between LDP peers
- Monitoring hello adjacencies in the network
- Gathering statistics regarding LDP sessions

LDP Entities

An LDP entity is uniquely identified by an LDP identifier that consists of the mplsLdpEntityLdpId and the mplsLdpEntityIndex (see the figure below) objects:

- The mplsLdpEntityLdpId consists of the local LSR ID (four octets) and the label space ID (two octets). The label space ID identifies a specific label space available within the LSR.
- The mplsLdpEntityIndex consists of the IP address of the peer active hello adjacency, which is the 32-bit representation of the IP address assigned to the peer LSR.

The mplsLdpEntityProtocolVersion is a sample object from the mplsLdpEntityTable.

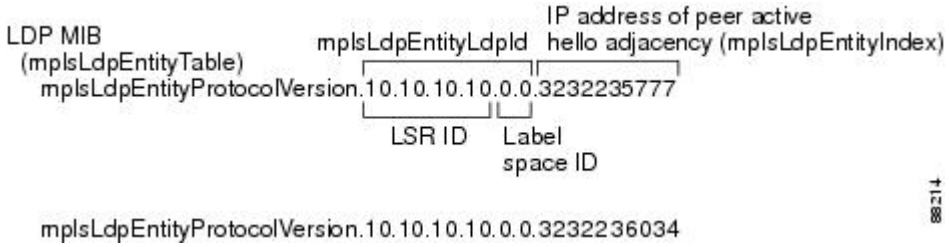
The figure below shows the following indexing:

- mplsLdpEntityLdpId = 10.10.10.10.0.0
- LSR ID = 10.10.10.10
- Label space ID = 0.0

Note The mplsLdpEntityLdpId or the LDP ID consists of the LSR ID and the label space ID.

- The IP address of peer active hello adjacency or the mplsLdpEntityIndex = 3232235777, which is the 32-bit representation of the IP address assigned to the peer's active hello adjacency.

Figure 1: Sample Indexing for an LDP Entity

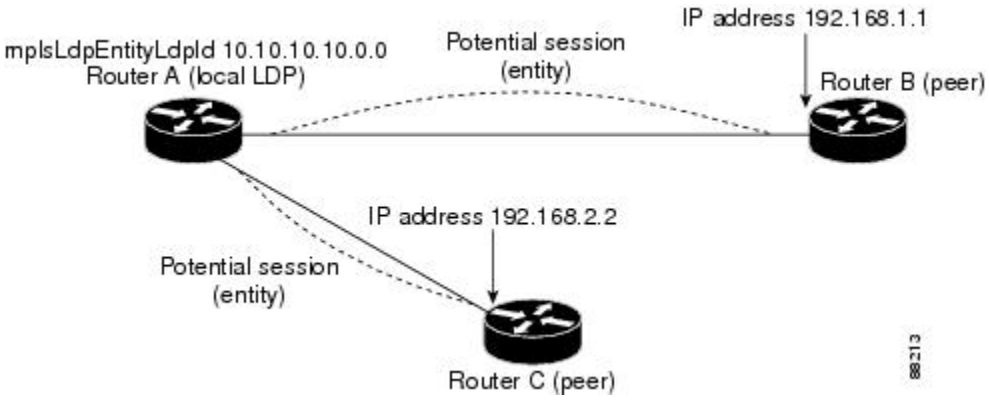


An LDP entity represents a label space that has the potential for a session with an LDP peer. An LDP entity is configured when a hello adjacency receives a hello message from an LDP peer.

In the figure below, Router A has potential sessions with two remote peers, Routers B and C. The mplsLdpEntityLdpId is 10.10.10.10.0.0, and the IP address of the peer active hello adjacency

(mplsLdpEntityIndex) is 3232235777, which is the 32-bit representation of the IP address 192.168.1.1 for Router B.

Figure 2: LDP Entity



LDP Sessions and Peers

LDP sessions exist between local entities and remote peers for the purpose of distributing label bindings. There is always a one-to-one correspondence between an LDP peer and an LDP session. A single LDP session is an LDP instance that communicates across one or more network links with a single LDP peer.

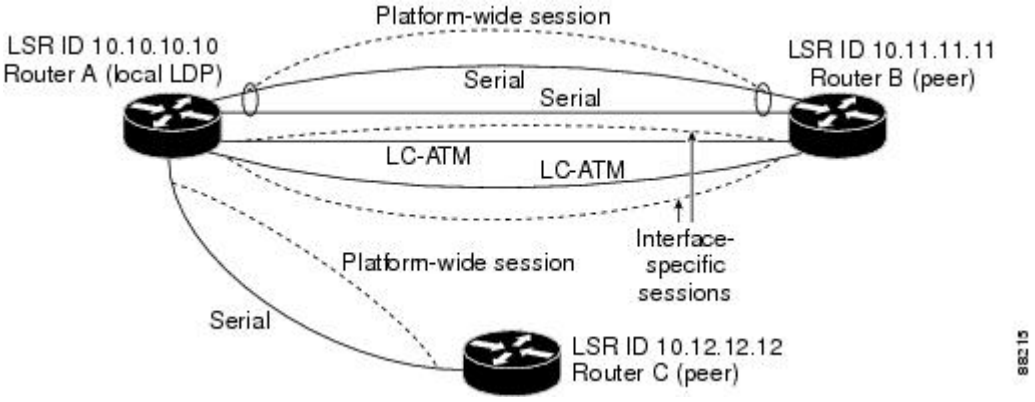
LDP supports the following types of sessions:

- Interface-specific—An interface-specific session uses interface resources for label space distributions. For example, each label-controlled ATM (LC-ATM) interface uses its own VPIs and VCIs for label space distributions. Depending on its configuration, an LDP platform can support zero, one, or more interface-specific sessions. Each LC-ATM interface has its own interface-specific label space and a nonzero label space ID.
- Platform-wide—An LDP platform supports a single platform-wide session for use by all interfaces that can share the same global label space. For Cisco platforms, all interface types except LC-ATM use the platform-wide session and have a label space ID of zero.

When a session is established between two peers, entries are created in the mplsLdpPeerTable and the mplsLdpSessionTable because they have the same indexing.

In the figure below, Router A has two remote peers, Routers B and C. Router A has a single platform-wide session that consists of two serial interfaces with Router B and another platform-wide session with Router C. Router A also has two interface-specific sessions with Router B.

Figure 3: LDP Sessions



The figure below shows entries that correspond to the mplsLdpPeerTable and the mplsLdpSessionTable in the figure above.

In the figure below, mplsLdpSesState is a sample object from the mplsLdpSessionTable on Router A. Four mplsLdpSesState sample objects are shown (top to bottom). The first object represents a platform-wide session associated with two serial interfaces. The next two objects represent interface-specific sessions for the LC-ATM interfaces on Routers A and B. These interface-specific sessions have nonzero peer label space IDs. The last object represents a platform-wide session for the next peer, Router C.

The indexing is based on the entries in the mplsLdpEntityTable. It begins with the indexes of the mplsLdpEntityTable and adds the following:

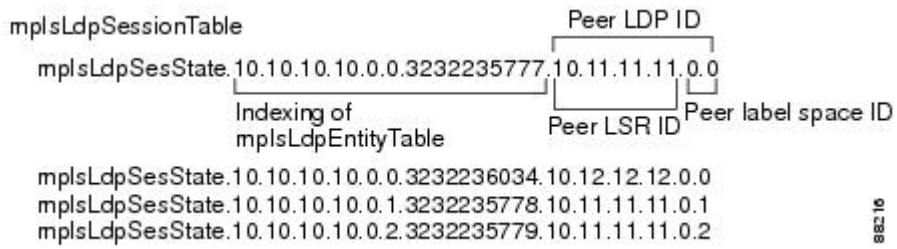
- Peer LDP ID = 10.11.11.11.0.0

The peer LDP ID consists of the peer LSR ID (four octets) and the peer label space ID (two octets).

- Peer LSR ID = 10.11.11.11
- Peer label space ID = 0.0

The peer label space ID identifies a specific peer label space available within the LSR.

Figure 4: Sample Indexing for an LDP Session



LDP Hello Adjacencies

An LDP hello adjacency is an association between a remotely discovered LDP process and a specific network path to reach the remote LDP process. An LDP hello adjacency enables two adjacent peers to exchange label binding information.

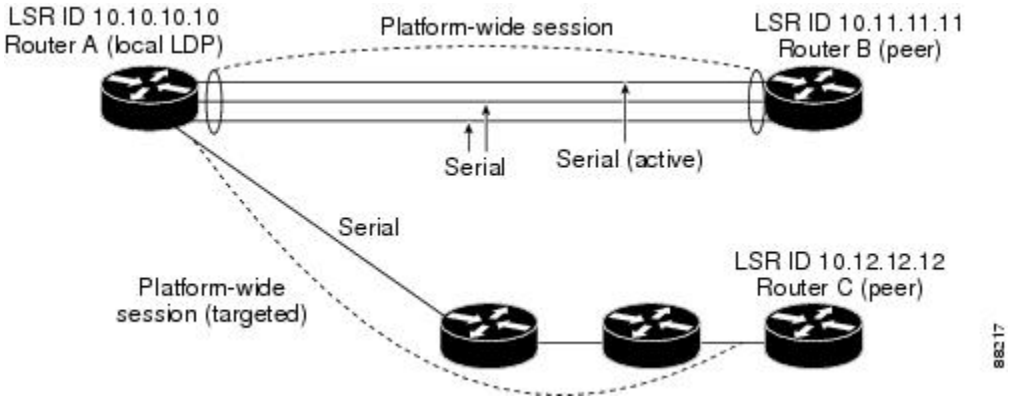
An LDP hello adjacency exists for each link on which LDP runs. Multiple LDP hello adjacencies exist whenever there is more than one link in a session between a router and its peer, such as in a platform-wide session.

A hello adjacency is considered active if it is currently engaged in a session, or nonactive if it is not currently engaged in a session.

A targeted hello adjacency is not directly connected to its peer and has an unlimited number of hops between itself and its peer. A linked hello adjacency is directly connected between two routers.

In the figure below, Router A has two remote peers, Routers B and C. Router A has a platform-wide session with Router B that consists of three serial interfaces, one of which is active and another platform-wide (targeted) session with Router C.

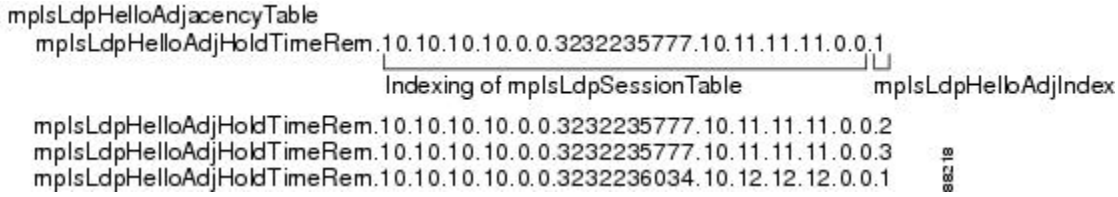
Figure 5: Hello Adjacency



The figure below shows entries in the mplsLdpHelloAdjacencyTable. There are four mplsLdpHelloAdjHoldTime sample objects (top to bottom). They represent the two platform-wide sessions and the four serial links shown in the figure above.

The indexing is based on the mplsLdpSessionTable. When the mplsLdpHelloAdjIndex enumerates the different links within a single session, the active link is mplsLdpHelloAdjIndex = 1.

Figure 6: Sample Indexing for an LDP Hello Adjacency



Events Generating MPLS LDP MIB Notifications

When you enable MPLS LDP MIB notification functionality by issuing the **snmp-server enable traps mpls rfc ldp** command, notification messages are generated and sent to a designated NMS in the network to signal the occurrence of specific events within Cisco IOS software.

The MPLS LDP MIB objects that announce LDP status transitions and event notifications are the following:

- mplsLdpSessionUp—This message is generated when an LDP entity (a local LSR) establishes an LDP session with another LDP entity (an adjacent LDP peer in the network). Enable this notification with the **session-up** keyword.
- mplsLdpSessionDown—This message is generated when an LDP session between a local LSR and its adjacent LDP peer is terminated. Enable this notification with the **session-down** keyword.

The up and down notifications indicate the last active interface in the LDP session.

- mplsLdpPathVectorLimitMismatch—This message is generated when a local LSR establishes an LDP session with its adjacent peer LSR, but the two LSRs have dissimilar path vector limits. Enable this notification with the **pv-limit** keyword.

The value of the path vector limit can range from 0 to 255; a value of 0 indicates that loop detection is off; any value other than 0 up to 255 indicates that loop detection is on and, in addition, specifies the maximum number of hops through which an LDP message can pass before a loop condition in the network is sensed.

We recommend that all LDP-enabled routers in the network be configured with the same path vector limit. Accordingly, the mplsLdpPathVectorLimitMismatch object exists in the MPLS LDP MIB to provide a warning message to the NMS when two routers engaged in LDP operations have a dissimilar path vector limits.

Note This notification is generated only if the distribution method is downstream-on-demand.

- mplsLdpFailedInitSessionThresholdExceeded—This message is generated when a local LSR and an adjacent LDP peer attempt to configure an LDP session between them, but fail to do so after a specified number of attempts. The default number of attempts is eight. This default value is implemented in Cisco IOS software and cannot be changed by either the CLI or an SNMP agent. Enable this notification with the **threshold** keyword.

Eight failed attempts to establish an LDP session between a local LSR and an LDP peer, due to any type of incompatibility between the devices, causes this notification message to be generated.

In general, Cisco routers support the same features across multiple platforms. Therefore, the most likely incompatibility to occur between Cisco LSRs is a mismatch of their respective ATM VPI and VCI label ranges.

For example, if you specify a range of valid labels for an LSR that does not overlap the range of its adjacent LDP peer, the routers try eight times to create an LDP session between themselves before the mplsLdpFailedInitSessionThresholdExceeded notification is generated and sent to the NMS as an informational message.

Occasionally, the LSRs whose label ranges do not overlap continue their attempt to create an LDP session between themselves after the eight retry limit is exceeded. In such cases, the LDP threshold exceeded

notification alerts the network administrator to the existence of a condition in the network that may warrant attention.

Note An mplsLdpEntityFailedInitSessionThreshold trap is supported only on an LC-ATM.

RFC 3036, *LDP Specification* , details the incompatibilities that can exist between Cisco routers or between Cisco routers and other vendor LSRs in an MPLS network. Among such incompatibilities, for example, are the following:

- • Nonoverlapping ATM VPI and VCI ranges (as previously noted) or nonoverlapping Frame Relay data-link connection identifiers (DLCI) ranges between LSRs attempting to configure an LDP session
  - Unsupported label distribution method
  - Dissimilar protocol data unit (PDU) sizes
  - Dissimilar LDP feature support

Scalar Objects in the MPLS LDP MIB Modules (RFC 3815)

The MPLS LDP MIB modules define several scalar objects. The table below describes the scalar objects that are implemented for Cisco IOS Release12.2(33)SRB.

Table 1: MPLS LDP MIB Scalar Objects and Descriptions

Object	Description
mplsLpdLsrId	The LSR's identifier. This is a globally unique value, such as the 32-bit router ID assigned to the LSR.
mplsLdpLsrLoopDetectionCapable	Loop detection capability of the LSR. Loop detection values are: none(1), other(2), hopCount(3), pathVector(4), and hopCountAndPathVector(5). The other(2) value indicates that the LSR supports loop detection, but does not support the three methods associated with values (3), (4), and (5).
mplsLdpEntityLastChange	The value of sysUpTime at the time of the most recent addition or deletion of an entry to or from the mplsLdpEntityTable or mplsLdpEntityStatsTable, or the most recent change in the value of any object in the mplsLdpEntityTable.
mplsLdpEntityIndexNext	Value to use for the mplsLdpEntityIndex when the router creates entries in the mplsLdpEntityTable. The value 0 indicates that no unassigned entries are available.

Object	Description
mplsLdpPeerLastChange	The value of sysUpTime at the time of the most recent addition or deletion to or from the mplsLdpPeerTable or mplsLdpSessionTable.

MIB Tables in the MPLS-LDP-STD-MIB Module (RFC 3815)

The MPLS-LDP-STD-MIB consists of the following tables. These tables define objects that are common to all LDP implementations.

- mplsLdpEntityTable (see the first table below)—Contains entries for every active LDP hello adjacency. Active and nonactive hello adjacencies appear in the mplsLdpHelloAdjacencyTable, rather than this table. This table is indexed by the local LDP identifier for the interface and the IP address of the peer active hello adjacency. (See the first figure above.)

The advantage of showing the active hello adjacency instead of sessions in this table is that the active hello adjacency can exist even if an LDP session is not active (cannot be established).

Directed adjacencies are also shown in this table. Associated adjacencies disappear when the targeted LDP session fails. Nondirected adjacencies might disappear from the mplsLdpEntityTable on some occasions, because adjacencies are deleted if the underlying interface becomes operationally down, for example.

- mplsLdpEntityStatsTable (see the second table below)—Augments the mplsLdpEntityTable and shares the same indexing for performing SNMP GETand GETNEXT operations. This table shows additional statistics for entities.
- mplsLdpPeerTable (see the third table below)—Contains entries for all peer sessions. This table is indexed by the local LDP identifier of the session, the IP address of the peer active hello adjacency, and the peer's LDP identifier. (See the fourth figure above.)
- mplsLdpSessionTable (see the fourth table below)—Augments the mplsLdpPeerTable and shares the same indexing for performing GET and GETNEXT operations. This table shows all sessions.
- mplsLdpSessionStatsTable (see the fifth table below)—Augments the mplsLdpPeerTable and shares the exact same indexing for performing GET and GETNEXT operations. This table shows additional statistics for sessions.
- mplsLdpHelloAdjacencyTable (see the sixth table below)—Contains entries for active and nonactive hello adjacencies. This table is indexed by the local LDP identifier of the associated session, the IP address of the peer active hello adjacency, the LDP identifier for the peer, and an arbitrary index that is set to the list position of the adjacency. (See the sixth figure above.)

MPLS LDP Entity Table (mplsLdpEntityTable) Objects and Descriptions

The table below describes the mplsLdpEntityTable objects.

Table 2: mplsLdpEntityTable Objects and Descriptions

Object	Description
mplsLdpEntityEntry	An LDP entity is a potential session between two peers.
mplsLdpEntityLdpId	The LDP identifier (not accessible) consists of the local LSR ID (four octets) and the label space ID (two octets).
mplsLdpEntityIndex	A secondary index that identifies this row uniquely. It consists of the IP address of the peer active hello adjacency, which is the 32-bit representation of the IP address assigned to the LSR (not accessible).
mplsLdpEntityProtocolVersion	The version number of the LDP protocol to be used in the session initialization message.
mplsLdpEntityAdminStatus	This is the administrative status of this LDP entity, which is always up. If the hello adjacency fails, this entity disappears from the mplsLdpEntityTable.
mplsLdpEntityOperStatus	This is the operational status of this LDP entity. Values are unknown(1), enabled(2), and disabled(3).
mplsLdpEntityTcpPort	This is the TCP discovery port for LDP or Tag Distribution Protocol (TDP). The default value is 646 (LDP).
mplsLdpEntityUdpDscPort	This is the User Datagram Protocol (UDP) discovery port for LDP or TDP. The default value is 646 (LDP).
mplsLdpEntityMaxPduLength	This is the maximum PDU length that is sent in the common session parameters of an initialization message.
mplsLdpEntityKeepAliveHoldTimer	The two-octet value that is the proposed keepalive hold time for this LDP entity.
mplsLdpEntityHelloHoldTimer	The two-octet value that is the proposed hello hold time for this LDP entity.
mplsLdpEntityInitSessionThreshold	The threshold for notification when this entity and its peer are engaged in an endless sequence of initialization messages. • The default value is 8 and cannot be changed by SNMP or the CLI.

Object	Description
mplsLdpEntityLabelDistMethod	The specified method of label distribution for any given LDP session. Values are downstreamOnDemand(1) and downstreamUnsolicited(2).
mplsLdpEntityLabelRetentionMode	Can be configured to use either conservative(1) for an LC-ATM or liberal(2) for all other interfaces.
mplsLdpEntityPathVectorLimit	If the value of this object is 0, loop detection for path vectors is disabled. Otherwise, if this object has a value greater than zero, loop detection for path vectors is enabled, and the path vector limit is this value. **Note** The mplsLdpEntityPathVectorLimit object is nonzero only if the mplsLdpEntityLabelDistMethod is downstreamOnDemand(1).
mplsLdpEntityHopCountLimit	If the value of this object is 0, loop detection using hop counters is disabled. • If the value of this object is greater than 0, loop detection using hop counters is enabled, and this object specifies this entity's maximum allowable value for the hop count. **Note** The mplsLdpEntityHopCountLimit object is nonzero only if the mplsLdpEntityLabelDistMethod is downstreamOnDemand(1).
mplsLdpEntityTransportAddrKind	If this value is interface(1), the IP address of the interface from the hello message is used as the transport address in the hello message. If this value is loopback(2), the IP address of the loopback interface is used as the address in the hello message.
mplsLdpEntityTargetPeer	If this LDP entity uses a targeted adjacency, this object is set to true(1). The default value is false(2).
mplsLdpEntityTargetPeerAddrType	The type of the internetwork layer address used for the extended discovery. This object indicates how the value of mplsLdpEntityTargPeerAddr is to be interpreted, as either IPv4 or IPv6.
mplsLdpEntityTargetPeerAddr	The value of the internetwork layer address used for the targeted adjacency.

Object	Description
mplsLdpEntityLabelType	Specifies the optional parameters for the LDP initialization message. If the value is generic(1), no optional parameters are sent in the LDP initialization message associated with this entity. • An LC-ATM uses atmParameters(2) to specify that a row in the mplsLdpEntityAtmParmsTable corresponds to this entry. **Note** Frame Relay parameters are not supported in Cisco IOS Release 12.2(33)SRB.
mplsLdpEntityDiscontinuityTime	The value of sysUpTime on the most recent occasion when one or more of this entity's counters suffered a discontinuity. The relevant counters are the specific instances of any Counter32 or Counter64 object contained in the mplsLdpEntityStatsTable that are associated with this entity. If no such discontinuities have occurred since the last reinitialization of the local management subsystem, this object contains a 0 value.
mplsLdpEntityStorageType	The storage type for this entry is a read-only implementation that is always volatile.
mplsLdpEntityRowStatus	This object is a read-only implementation that is always active.

MPLS LDP Entity Statistics Table (mplsLdpEntityStatsTable) Objects and Descriptions

The table below describes the mplsLdpEntityStatsTable objects.

Table 3: mplsLdpEntityStatsTable Objects and Descriptions

Object	Description
mplsLdpEntityStatsEntry	These entries augment the mplsLdpEntityTable by providing additional information for each entry.
mplsLdpEntityStatsSessionsAttempts	Not supported in Cisco IOS Release 12.2(33)SRB.
mplsLdpEntityStatsSessionRejectedNoHelloErrors	A count of the session rejected no hello error notification messages sent or received by this LDP entity.
mplsLdpEntityStatsSessionRejectedAdErrors	A count of the session rejected parameters advertisement mode error notification messages sent or received by this LDP entity.

Object	Description
mplsLdpEntityStatsSessionRejectedMaxPduErrors	A count of the session rejected parameters max PDU length error notification messages sent or received by this LDP entity.
mplsLdpEntityStatsSessionRejectedLRErrors	A count of the session rejected parameters label range notification messages sent or received by this LDP entity.
mplsLdpEntityStatsBadLdpIdentifierErrors	A count of the number of bad LDP identifier fatal errors detected by the session associated with this LDP entity.
mplsLdpEntityStatsBadPduLengthErrors	A count of the number of bad PDU length fatal errors detected by the session associated with this LDP entity.
mplsLdpEntityStatsBadMessageLengthErrors	A count of the number of bad message length fatal errors detected by the session associated with this LDP entity.
mplsLdpEntityStatsBadTlvLengthErrors	A count of the number of bad type, length, values (TLVs) length fatal errors detected by the session associated with this LDP entity.
mplsLdpEntityStatsMalformedTlvValueErrors	A count of the number of malformed TLV value fatal errors detected by the session associated with this LDP entity.
mplsLdpEntityStatsKeepAliveTimerExpErrors	A count of the number of session keepalive timer expired errors detected by the session associated with this LDP entity.
mplsLdpEntityStatsShutdownReceivedNotifications	A count of the number of shutdown notifications received related to the session associated with this LDP entity.
mplsLdpEntityStatsShutdownSentNotifications	A count of the number of shutdown notifications sent related to the session associated with this LDP entity.

MPLS LDP Peer Table (mplsLdpPeerTable) Objects and Descriptions

The table below describes the mplsLdpPeerTable objects.

Table 4: mplsLdpPeerTable Objects and Descriptions

Object	Description
mplsLdpPeerEntry	Information about a single peer that is related to a session (not accessible). **Note** This table is augmented by the mplsLdpSessionTable.
mplsLdpPeerLdpId	The LDP identifier of this LDP peer (not accessible) consists of the peer LSR ID (four octets) and the peer label space ID (two octets).
mplsLdpPeerLabelDistMethod	For any given LDP session, the method of label distribution. Values are downstreamOnDemand(1) and downstreamUnsolicited(2).
mplsLdpPeerPathVectorLimit	If the value of mplsLdpPeerLabelDistMethod is downstreamOnDemand (1), this object represents the path vector limit for this peer. If the value of the mplsLdpPeerLabelDistMethod object is downstreamUnsolicited (2), this value should be 0.
mplsLdpPeerTransportAddrType	Type of Internet address for the mplsLdpPeerTransportAddr object—either the IPv4 transport address or IPv6 transport address used in the opening TCP session or the IPv4 or IPv6 source address for the UDP carrying the hello messages.
mplsLdpPeerTransportAddr	Internet address advertised by the peer in the hello message or the hello source address specified by the mplsLdpPeerTransportAddrType object.

MPLS LDP Session Table (mplsLdpSessionTable) Objects and Descriptions

The table below describes the mplsLdpSessionTable objects.

Table 5: mplsLdpSessionTable Objects and Descriptions

Object	Description
mplsLdpSessionEntry	An entry in this table represents information on a single session between an LDP entity and an LDP peer. The information contained in a row is read-only. This table augments the mplsLdpPeerTable.

Object	Description
mplsLdpSessionStateLastChange	The value of sysUpTime at the time the session entered its state denoted by the mplsLdpSessionState object.
mplsLdpSessionState	The current state of the session. All of the states are based on the LDP or TDP state machine for session negotiation behavior. The states are as follows: • nonexistent(1) • initialized(2) • openrec(3) • opensent(4) • operational(5)
mplsLdpSessionRole	The value of this object indicates whether the LSR or LER takes an active(2) or passive(3) role when a session is established. If the role of the LSR or LER cannot be determined, the value of the object is unknown(1).
mplsLdpSessionProtocolVersion	The version of the LDP protocol that this session is using. This is the version of the LDP protocol that has been negotiated during session initialization.
mplsLdpSessionKeepAliveHoldTimeRem	The keepalive hold time remaining for this session.
mplsLdpSessionKeepAliveTime	The time in seconds between keepalive messages negotiated between a configured value and the peer's proposed keepalive hold timer value. The value is the lower of the two.
mplsLdpSessionMaxPduLen	The value of maximum allowable length for LDP PDUs for this session. This value could have been negotiated during the session initialization.
mplsLdpSessionDiscontinuityTime	The value of sysUpTime on the most recent occasion when one or more of this session's counters suffered a discontinuity. The relevant counters are the specific instances of any Counter32 or Counter64 object contained in the mplsLdpSesStatsTable associated with this session. The initial value of this object is the value of sysUpTime when the entry was created in this table.

MPLS LDP Session Statistics Table (mplsLdpSessionStatsTable) Objects and Descriptions

The table below describes the mplsLdpSessionStatsTable objects.

Table 6: mplsLdpSessionStatsTable Objects and Descriptions

Object	Description
mplsLdpSessionStatsEntry	An entry in this table represents statistical information on a single session between an LDP entity and an LDP peer. This table augments the mplsLdpPeerTable.
mplsLdpSessionStatsUnknownMesTypeErrors	This object is the count of the number of unknown message type errors detected during this session.
mplsLdpSessionStatsUnknownTlvErrors	This object is the count of the number of unknown TLV errors detected during this session.

MPLS LDP Hello Adjacency Table (mplsLdpHelloAdjacencyTable) Objects and Descriptions

The table below describes the mplsLdpHelloAdjacencyTable objects.

Table 7: mplsLdpHelloAdjacencyTable Objects and Descriptions

Object	Description
mplsLdpHelloAdjacencyEntry	Each row represents a single LDP hello adjacency. An LDP session can have one or more hello adjacencies (not accessible).
mplsLdpHelloAdjacencyIndex	An identifier for this specific adjacency (not accessible). The active hello adjacency has the mplsLdpHelloAdjIndex object equal to 1.
mplsLdpHelloAdjacencyHoldTimeRem	The time remaining for this hello adjacency. This interval changes when the next hello message, which corresponds to this hello adjacency, is received.
mplsLdpHelloAdjacencyHoldTime	The hello time negotiated between the LSR or LER and its peer. If this value is 0, the defaults are used, 15 seconds for link hellos and 45 seconds for targeted hellos. If this value is 65535, the hold time is infinite.
mplsLdpHelloAdjacencyType	This adjacency is the result of a link hello if the value of this object is link(1). Otherwise, this adjacency is a result of a targeted hello and its value is targeted(2).

MIB Tables in the MPLS-LDP-ATM-STD-MIB Module (RFC 3815)

The MPLS-LDP-ATM-STD-MIB consists of the following tables. These tables define Layer 2 ATM-related objects for use with the MPLS-LDP-STD-MIB.

- mplsLdpEntityAtmTable (see the first table below)—Contains entries for every LDP-enabled LC-ATM interface. This table is indexed in the same way as the mplsLdpEntityTable although only LC-ATM interfaces are shown.
- mplsLdpEntityAtmLRTable (see the second table below)—Contains entries for every LDP-enabled LC-ATM interface. Indexing is the same as it is for the mplsLdpEntityTable, except two indexes have been added, mplsLdpEntityAtmLRMinVpi and mplsLdpEntityAtmLRMinVci. These additional indexes allow more than one label range to be defined. However, in the Cisco IOS Release 12.2(33)SRB implementation, only one label range per LC-ATM interface is allowed.
- mplsLdpAtmSessionTable (see the third table below)—Contains entries for LDP-enabled LC-ATM sessions. Indexing is the same as it is for the mplsLdpPeerTable, except two indexes have been added, mplsLdpAtmSessionLRLowerBoundVpi and mplsLdpAtmSessionLRLowerBoundVci. These additional indexes allow more than one label range to be defined. However, in the Cisco IOS Release 12.2(33)SRB implementation, only one label range per LC-ATM interface is allowed.

MPLS LDP Entity ATM Table (mplsLdpEntityAtmTable) Objects and Descriptions

The table below describes the mplsLdpEntityAtmTable objects.

Table 8: mplsLdpEntityAtmTable Objects and Descriptions

Object	Description
mplsLdpEntityAtmEntry	Represents the ATM parameters and ATM information for this LDP entity.
mplsLdpEntityAtmIfIndxOrZero	This value represents the SNMP IF-MIB index for the interface-specific LC-ATM entity.
mplsLdpEntityAtmMergeCap	Denotes the merge capability of this entity.
mplsLdpEntityAtmLRComponents	Number of label range components in the initialization message. This also represents the number of entries in the mplsLdpEntityConfAtmLRTable that correspond to this entry. **Note** Cisco IOS software supports only one component.

Object	Description
mplsLdpEntityAtmVcDirectionality	If the value of this object is bidirectional(0), a given VCI within a given VPI is used as a label for both directions independently of one another. If the value of this object is unidirectional(1), a given VCI within a VPI designates one direction.
mplsLdpEntityAtmLsrConnectivity	The peer LSR can be connected indirectly by means of an ATM VPI, so that the VPI values can be different on the endpoints. For that reason, the label must be encoded entirely within the VCI field. Values are direct(1) and indirect(2). The default is direct(1).
mplsLdpEntityAtmDefaultControlVpi	The default VPI value for the non-MPLS connection.
mplsLdpEntityAtmDefaultControlVci	The default VCI value for the non-MPLS connection.
mplsLdpEntityAtmUnlabTrafVpi	VPI value of the virtual channel connector (VCC) supporting unlabeled traffic. This non-MPLS connection is used to carry unlabeled (IP) packets.
mplsLdpEntityAtmUnlabTrafVci	VCI value of the VCC supporting unlabeled traffic. This non-MPLS connection is used to carry unlabeled (IP) packets.
mplsLdpEntityAtmStorageType	The storage type for this entry is a read-only implementation that is always volatile.
mplsLdpEntityAtmRowStatus	This object is a read-only implementation that is always active.

MPLS LDP Entity ATM Label Range Table (mplsLdpEntityAtmLRTable) Objects and Descriptions

The table below describes the mplsLdpEntityAtmLRTable objects.

Table 9: mplsLdpEntityAtmLRTable Objects and Descriptions

Object	Description
mplsLdpEntityAtmLREntry	A row in the LDP Entity Configurable ATM Label Range Table. One entry in this table contains information on a single range of labels; the range is defined by an upper boundary (VPI/VCI pair) and a lower boundary (VPI/VCI pair). This is the same data used in the initialization message. This label range should overlap the label range of the peer.

Object	Description
mplsLdpEntityAtmLRMinVpi	The minimum VPI number configured for this range (not accessible).
mplsLdpEntityAtmLRMinVci	The minimum VCI number configured for this range (not accessible).
mplsLdpEntityAtmLRMaxVpi	The maximum VPI number configured for this range (not accessible).
mplsLdpEntityAtmLRMaxVci	The maximum VCI number configured for this range (not accessible).
mplsLdpEntityAtmLRStorageType	The storage type for this entry is a read-only implementation that is always volatile.
mplsLdpEntityAtmLRRowStatus	This object is a read-only implementation that is always active.

MPLS LDP ATM Session Table (mplsLdpAtmSessionTable) Objects and Descriptions

The table below describes the mplsLdpAtmSessionTable objects.

Table 10: mplsLdpAtmSessionTable Objects and Descriptions

Objects	Description
mplsLdpAtmSessionEntry	An entry in this table represents information on a single label range intersection between an LDP entity and an LDP peer (not accessible).
mplsLdpAtmSessionLRLowerBoundVpi	The minimum VPI number configured for this range (not accessible).
mplsLdpAtmSessionLRLowerBoundVci	The minimum VCI number configured for this range (not accessible).
mplsLdpAtmSessionLRUpperBoundVpi	The maximum VPI number configured for this range (read-only).
mplsLdpAtmSessionLRUpperBoundVci	The maximum VCI number configured for this range (read-only).

MIB Table in the MPLS-LDP-GENERIC-STD-MIB Module (RFC 3815)

The MPLS-LDP-GENERIC-STD-MIB contains the following table. This table defines Layer 2 per-platform objects for use with the MPLS-LDP-STD-MIB.

- mplsLdpEntityGenericLRTable (see the table below)—Contains entries for every LDP-enabled interface that is in the global label space. (For Cisco, this applies to all interfaces except LC-ATM. LC-ATM entities are shown in the mplsLdpEntityAtmLRTable instead.) Indexing is the same as it is for the mplsLdpEntityTable, except two indexes have been added, mplsLdpEntityGenericLRMin and mplsLdpEntityGenericLRMax. These additional indexes allow more than one label range to be defined. However, in the Cisco IOS Release 12.2(33)SRB implementation, only one global label range is allowed.

MPLS LDP Entity Generic Label Range Table (mplsLdpEntityGenericLRTable) Objects and Descriptions

The table below describes the mplsLdpEntityGenericLRTable objects.

Table 11: mplsLdpEntityGenericLRTable Objects and Descriptions

Object	Description
mplsLdpEntityGenericLREntry	A row in the LDP Entity Configurable Generic Label Range table. One entry in this table contains information on a single range of labels; the range is defined by an upper boundary and a lower boundary. • The current implementation supports one label range per entity.
mplsLdpEntityGenericLRMin	The minimum label configured for this range (not accessible).
mplsLdpEntityGenericLRMax	The maximum label configured for this range (not accessible).
mplsLdpEntityGenericLabelSpace	This value indicates whether the label space type is perPlatform (1) or perInterface (2).
mplsLdpEntityGenericIfIndxOrZero	This value represents the SNMP IF-MIB index for the platform-wide entity. If the active hello adjacency is targeted, the value is 0.
mplsLdpEntityGenericLRStorageType	The storage type for this entry is a read-only implementation that is always volatile.
mplsLdpEntityGenericLRRowStatus	This object is a read-only implementation that is always active.

VPN Contexts in the MPLS LDP MIB

Within an MPLS Border Gateway Protocol (BGP) 4 Virtual Private Network (VPN) environment, separate LDP processes can be created for each VPN. These processes and their associated data are called LDP contexts. Each context is independent from all others and contains data specific only to that context.

The VPN Aware LDP MIB feature enables the LDP MIB to get VPN context information. The feature adds support for different contexts for different MPLS VPNs. Users of the MIB can display MPLS LDP processes for a given MPLS VPN. The VPN Aware LDP MIB feature does not change the syntax of the MPLS LDP MIB. It changes the number and types of entries within the tables.

The MPLS LDP MIB can show information about only one context at a time. With Cisco IOS Release 12.2(33)SRB, you can specify a context—either a global context or an MPLS VPN context—using an SNMP security name.

The following sections describe topics related to the VPN Aware LDP MIB feature:

SNMP Contexts

SNMP contexts provide VPN users with a secure way of accessing MIB data. When a VPN is associated with a context, that VPN's specific MIB data exists in that context. Associating a VPN with a context enables service providers to manage networks with multiple VPNs. Creating and associating a context with a VPN enables a provider to prevent the users of one VPN from accessing information about users of other VPNs on the same networking device.

VPN-aware SNMP requires that SNMP manager and agent entities operating in a VPN environment agree on mapping between the SNMP security name and the VPN name. This mapping is created by you using different contexts for the SNMP data of different VPNs, which is accomplished through the configuration of the SNMP View-based Access Control Model MIB (SNMP-VACM-MIB). The SNMP-VACM-MIB is configured with views so that a user on a VPN with a security name is allowed access to the restricted object space within the context of only that VPN.

SNMP request messages undergo three phases of security and access control before a response message is sent back with the object values within a VPN context:

- The first security phase is authentication of the username. During this phase, the user is authorized for SNMP access.
- The second phase is access control. During this phase, the user is authorized for SNMP access to the group objects in the requested SNMP context.
- In the third phase, the user can access a particular instance of a table entry. With this third phase, complete retrieval can be based on the SNMP context name.

IP access lists can be configured and associated with SNMP community strings. This feature enables you to configure an association between VPN routing and forwarding (VRF) instances and SNMP community strings. When a VRF instance is associated with an SNMP community string, SNMP processes requests coming in for a particular community string only if they are received from the configured VRF. If the community string contained in the incoming packet does not have a VRF associated with it, it is processed only if it came in through a non-VRF interface.

You can also enable or disable authentication traps for SNMP packets dropped due to VRF mismatches. By default, if SNMP authentication traps are enabled, VRF authentication traps are also enabled.

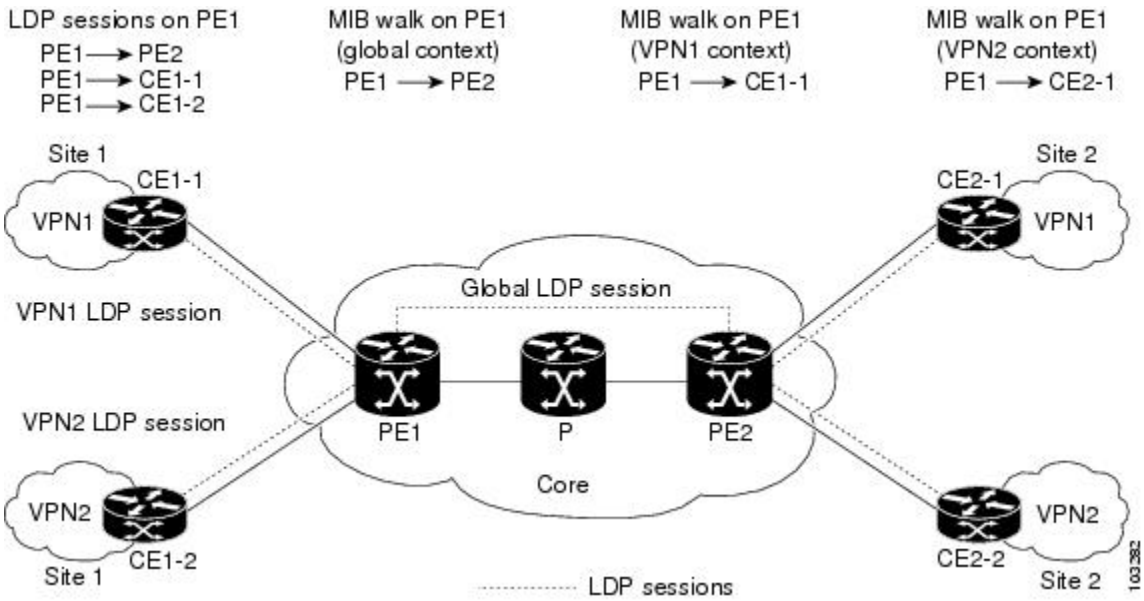
VPN Aware LDP MIB Sessions

The VPN Aware LDP MIB feature supports an SNMP query to the MPLS LDP MIB for both global and VPN contexts. This feature allows you to enter LDP queries on any VRF and on the core (global context). A query can differentiate between LDP sessions from different VPNs. LDP session information for a VPN stays in the context of that VPN. Therefore, the information from one VPN is not available to a user of a different VPN. The VPN Aware LDP MIB also allows you to display LDP processes operating in a Carrier Supporting Carrier (CSC) network.

In an MPLS VPN, a service provider edge router (PE) might contain VRFs for several VPNs and a global routing table. To set up separate LDP processes for different VPNs on the same device, you need to configure each VPN with a unique securityName, contextName, and View-based Access Control Model (VACM) view. The VPN securityName must be configured for the MPLS LDP MIB.

The figure below shows LDP sessions for a sample MPLS VPN with the VPN Aware LDP MIB feature.

Figure 7: MPLS LDP Sessions with the VPN Aware LDP MIB Feature



With the VPN Aware LDP MIB feature, you can do MIB queries or MIB walks for an MPLS VPN LDP session or a global LDP session.

Note To verify LDP session information for a specific VPN, use the show mpls ldp neighbor **vrf** *vpn-name* **detail** command.

VPN Aware LDP MIB Notifications

The VPN Aware LDP MIB feature supports LDP notifications for multiple LDP contexts for VPNs. LDP notifications can be generated for the core (global context) and for different VPNs. You can cause notifications be sent to different NMS hosts for different LDP contexts. LDP notifications associated with a specific VRF

are sent to the NMS designated for that VRF. LDP global notifications are sent to the NMS configured to receive global traps.

To enable LDP context notifications for the VPN Aware LDP MIB feature, use either the SNMP mplsLdpSessionsUpDownEnable object (in the global LDP context only) or the following extended global configuration commands.

To enable LDP notifications for the global context, use the following commands on a provider edge (PE) router:

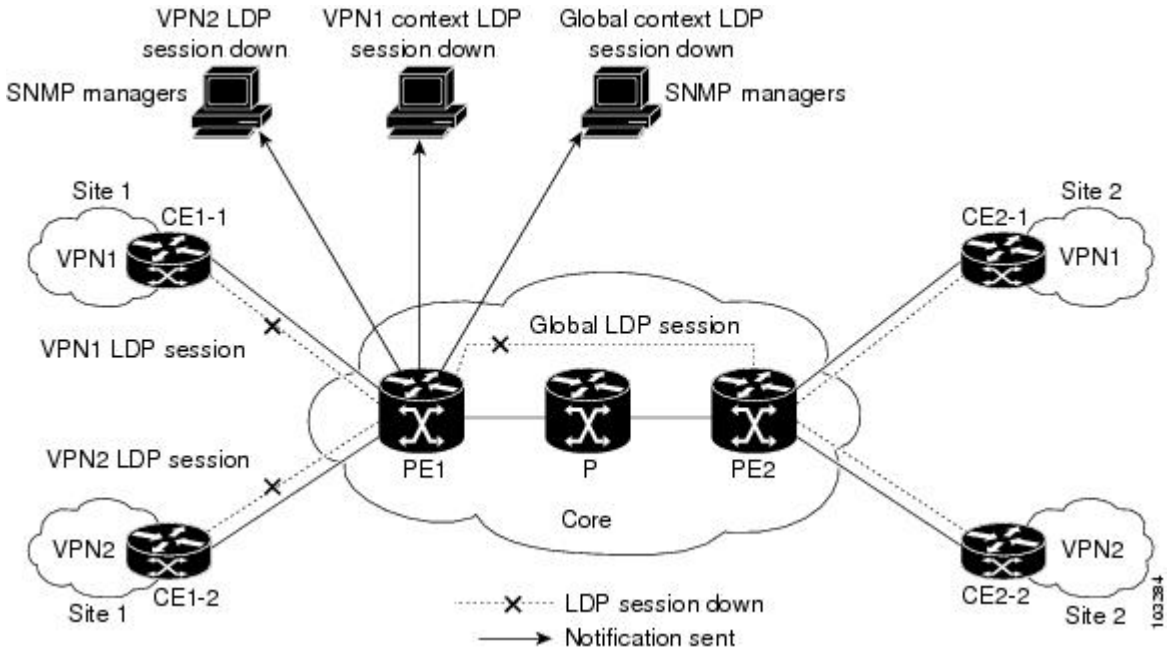
```
Router(config)# snmp-server host host-address trapscommunity
 mpls-ldp
Router(config)# snmp-server enable traps mpls rfc ldp
```

To enable LDP notifications for a VPN context, use the following commands:

```
Router(config)# snmp-server host host-address vrf
vrf-name version
 {v1
|v2c
|v3
community
 community-string
 udp-port
 upd-port
 mpls-ldp
Router(config)# snmp-server enable traps mpls rfc ldp
```

The figure below shows LDP notifications with the VPN Aware LDP MIB feature.

Figure 8: LDP Notifications with the VPN Aware LDP MIB Feature



Differences Between the MPLS-LDP-STD-MIB and the MPLS-LDP-MIB

The MPLS-LDP-STD-MIB based on RFC 3815 provides the same basic functionality as the MPLS-LDP-MIB based on the IETF Version 8 draft (draft-ieft-mpls-ldp-08.txt). The module identity was changed from mplsLdpMIB to mplsLdpStdMIB. Both MIBs provide an interface for managing LDP through the use of SNMP.

After the implementation of the MPLS-LDP-STD-MIB (RFC 3815) in Cisco IOS Release 12.2(33)SRB the MPLS-LDP-MIB will exist for a period of time before support is completely removed. This gives you the chance to migrate to the MPLS-LDP-STD-MIB. Both MIBs can coexist in the same image because the MPLS-LDP-STD-MIB and the MPLS-LDP-MIB have different root object identifiers (OIDs).

The following sections contain information about the major differences between the MPLS-LDP-STD-MIB and the MPLS-LDP-MIB:

MPLS-LDP-MIB and MPLS-LDP-STD-MIB Scalar Object Differences

The table below shows the difference between the scalar objects in the MPLS-LDP-MIB and the MPLS-LDP-STD-MIB.

Table 12: Scalar Objects: MPLS-LDP-MIB and MPLS-LDP-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
—	mplsLdpPeerLastChange	New Object—Indicates the value of the sysUpTime at the time of the most recent addition or deletion to or from the mplsLdpPeerTable or the mplsLdpSessionTable.
mplsLdpSesUpDownTrapEnable	—	Object deleted.
—	mplsFecLastChange **Note** Not supported in Cisco IOS Releases 12.2(33)SRB and 12.2(33)SB.	New object—Contains the sysUpTime at the time of the most recent change to the mplsLdpFecTable.
—	mplsLdpLspFecLastChange **Note** Not supported in Cisco IOS Releases 12.2(33)SRB and 12.2(33)SB.	New object—Stores the last change time for the mplsLdpLspFecTable.
—	mplsLdpEntityLastChange	New object—Indicates the last change time for the mplsLdpEntityTable or mplsLdpEntityStatsTable.

MPLS-LDP-MIB and MPLS-LDP-STD-MIB Table Object Differences

The following tables show the major differences between the MPLS-LDP-MIB and the MPLS-LDP-STD-MIB objects for each table.

MPLS LDP Entity Table (mplsLdpEntityTable) Differences

The table below shows the major differences between MPLS-LDP-MIB and MPLS-LDP-STD-MIB objects for the MPLS LDP entity table.

Table 13: MPLS LDP Entity Table: MPLS-LDP-MIB and MPLS-LDP-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpEntityIndexNext	mplsLdpEntityIndexNext	Syntax changed—Unsigned32 to IndexIntegerNextFree.
mplsLdpEntityIndex	mplsLdpEntityIndex	Syntax changed—Unsigned32 to IndexInteger.
mplsLdpEntityProtocolVersion	mplsLdpEntityProtocolVersion	Syntax changed—Unsigned32 to Integer32.
mplsLdpEntityAdminStatus	mplsLdpEntityAdminStatus	Description changed—Modified to suggest that an NMS clean up any related entry in the mplsLdpPeerTable in case this object is changed from enable to disable.
mplsLdpEntityOperStatus	mplsLdpEntityOperStatus	Description changed—Modified to include the value of unknown(1) for a transient condition before the LSR changes the value to enabled(2) or disabled(3).
mplsLdpEntityTcpDscPort	mplsLdpEntityTcpPort	Object name changed.
mplsLdpEntityMaxPduLength	mplsLdpEntityMaxPduLength	Syntax changed—Lower-limit value for this object increased from 0 to 256. Description changed—Indicates that the receiving LSR should calculate the maximum PDU length for the session by using the smaller of its and its peer's proposal for the Max PDU length.
mplsLdpEntityInitSessionThreshold	mplsLdpEntityInitSessionThreshold	Syntax changed—Maximum value changed from MAXINT to 100.
mplsLdpEntityLabelDistMethod	mplsLdpEntityLabelDistMethod	Syntax changed—INTEGER to MplsLabelDistributionMethod TC.

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpEntityLabelRetentionMode	mplsLdpEntityLabelRetentionMode	Syntax changed—INTEGER to MplsRetentionMode TC.
mplsLdpEntityPVLMisTrapEnable	—	Notification control object removed.
mplsLdpEntityPVL	mplsLdpEntityPathVectorLimit	Object name changed.
mplsLdpEntityHopCountLimit	mplsLdpEntityHopCountLimit	DEFVAL clause added for a default value = 0.
—	mplsLdpEntityTransportAddrKind	New object—Indicates the address to be used for hello messages (interface and loopback).
mplsLdpEntityTargPeerAddrType	mplsLdpEntityTargPeerAddrType	Syntax changed—To InetAddressType.
mplsLdpEntityTargPeerAddr	mplsLdpEntityTargPeerAddr	Syntax changed—To InetAddress.
mplsLdpEntityOptionalParameters	mplsLdpEntityLabelType	Object name changed.
mplsLdpEntityStorType	mplsLdpEntityStorageType	Object name changed.
mplsLdpEntityRowStatus	mplsLdpEntityRowStatus	Description change—Added recommendation to set the mplsLdpentityAdminStatus to down, change the objects in this entry, and then set the Admin status to enable.

MPLS LDP Entity Statistics Table (mplsLdpEntityStatsTable) Differences

Note A general paragraph regarding discontinuities is added to all the counter objects in MPLS LDP entity statistics table: "Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of mplsLdpEntityDiscontinuityTime."

The table below shows the difference between MPLS-LDP-MIB and MPLS-LDP-STD-MIB objects for the MPLS LDP entity statistics table.

Table 14: MPLS LDP Entity Statistics Table: MPLS-LDP-MIB and MPLS-LDP-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpAttemptedSessions	mplsLdpEntityStatsSessionAttempts	Object name changed.
mplsLdpSesRejectedNoHelloErrors	mplsLdpEntityStatsSessionRejectedNoHelloErrors	Object name changed.

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpSesRejectedAdErrors	mplsLdpEntityStatsSessionRejectedAdErrors	Object name changed.
mplsLdpSesRejectedMaxPduErrors	mplsLdpEntityStatsSessionRejectedMaxPduErrors	Object name changed.
mplsLdpSesRejectedLRErrors	mplsLdpEntityStatsSessionRejectedLRErrors	Object name changed.
mplsLdpBadLdpIdentifierErrors	mplsLdpEntityStatsBadLdpIdentifierErrors	Object name changed.
mplsLdpBadPduLengthErrors	mplsLdpEntityStatsBadPduLengthErrors	Object name changed.
mplsLdpBadMessageLengthErrors	mplsLdpEntityStatsBadMessageLengthErrors	Object name changed.
mplsLdpBadTlvLengthErrors	mplsLdpEntityStatsBadTlvLengthErrors	Object name changed.
mplsLdpMalformedTlvValueErrors	mplsLdpEntityStatsMalformedTlvValueErrors	Object name changed.
mplsLdpKeepAliveTimerExpErrors	mplsLdpEntityStatsKeepAliveTimerExpErrors	Object name changed.
mplsLdpShutdownNotifReceived	mplsLdpEntityStatsShutdownReceivedNotifications	Object name changed.
mplsLdpShutdownNotifSent	mplsLdpEntityStatsShutdownSentNotifications	Object name changed.

MPLS LDP Peer Table (mplsLdpPeerTable) Differences

The table below shows the difference between MPLS-LDP-MIB and MPLS-LDP-STD-MIB objects for the MPLS LDP peer table.

Table 15: MPLS LDP Peer Table: MPLS-LDP-MIB and MPLS-LDP-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpPeerLabelDistMethod	mplsLdpPeerLabelDistMethod	Syntax change—From INTEGER to MplsLabelDistributionMethod
mplsLdpPeerLoopDectionForPV	—	Object deleted.
mplsLdpPeerPVL	mplsLdpPeerPathVectorLimit	Object name changed.
—	mplsLdpPeerTransportAddrType	New object—Internet address type (IPv4 or IPv6) advertised by the peer in the hello message or hello source message.
—	mplsLdpPeerTransportAddr	New object—Internet address (IPv4 or IPv6) advertised by the peer in the hello message or hello source message.

MPLS LDP Session Table (mplsLdpSessionTable) Differences

The table below shows the difference between MPLS-LDP-MIB and MPLS-LDP-STD-MIB objects for the MPLS LDP session table.

Table 16: MPLS LDP Session Table: MPLS-LDP-MIB and MPLS-LDP-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpSesState	mplsLdpSessionState	Object name changed.
mplsLdpSesProtocolVersion	mplsLdpSessionProtocolVersion	Object name changed.
—	mplsLdpSessionStateLastChange	New Object—Indicates the last change time for this table.
—	mplsLdpSessionRole	New object—Indicates the LSR state: active, passive, or unknown.
mplsLdpSesKeepAliveHoldTimeRem	mplsLdpSessionKeepAliveHoldTimeRem	Object name changed.
—	mplsLdpSessionKeepAliveTime	New object—Indicates the actual keepalive time negotiated between peers.
mplsLdpSesMaxPduLen	mplsLdpSessionMaxPduLen	Object name changed.
mplsLdpSesDiscontinuityTime	mplsLdpSessionDiscontinuityTime	Object name changed.

MPLS LDP Hello Adjacency Table (mplsLdpHelloAdjacencyTable) Differences

The table below shows the difference between MPLS-LDP-MIB and MPLS-LDP-STD-MIB objects for the MPLS LDP hello adjacency table.

Table 17: MPLS LDP Hello Adjacency Table: MPLS-LDP-MIB and MPLS-LDP-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-STD-MIB Object	Difference
mplsLdpHelloAdjIndex	mplsLdpHelloAdjacencyIndex	Object name changed.
mplsLdpHelloAdjHoldTimeRem	mplsLdpHelloAdjacencyHoldTimeRem	Object name changed.
—	mplsLdpHelloAdjacencyHoldTime	New object—Indicates the actual negotiated adjacency hold time.
mplsLdpHelloAdjType	mplsLdpHelloAdjacencyType	Object name changed.

MPLS-LDP-MIB and MPLS-LDP-STD-MIB Notification Changes

All notifications from the MPLS-LDP-MIB are retained in the MPLS-LDP-STD-MIB. The following changes are implemented for the notifications in MPLS-LDP-STD-MIB:

- The mplsLdpPVLMismatch notification is renamed to mplsLdpPathVectorLimitMismatch.
- The mplsLdpEntityPVLMisTrapEnable, notification control object, was removed for the MPLS-LDP-STD-MIB.
- The Cisco IOS command for enabling notifications for the MPLS-LDP-MIB is different from the command for enabling notifications for the MPLS-LSR-STD-MIB.

For the MPLS-LDP-MIB, the command is **snmp-server enable traps mpls ldp**. For the MPLS-LSR-STD-MIB, the command is **snmp-server enable traps mpls rfc ldp**.

Differences Between the MPLS-LDP-MIB and the MPLS-LDP-ATM-STD-MIB (RFC 3815)

Layer 2 ATM-related objects are removed from the MPLS-LDP-MIB and placed in a new MIB module in RFC 3815. The new MIB module is the MPLS-LDP-ATM-STD-MIB. This action provides modularity and reduces the size of the MPLS LDP MIB.

The table below lists the differences between ATM-related objects in the MPLS-LDP-MIB and objects in the MPLS-LDP-ATM-STD-MIB.

Table 18: Differences Between MPLS-LDP-MIB and MPLS-LDP-ATM-STD-MIB Objects

MPLS-LDP-MIB Object	MPLS-LDP-ATM-STD-MIB Object	Difference
mplsLdpEntityAtmParmsTable	mplsLdpEntityAtmTable	Object name changed.
mplsLdpEntityAtmMergeCap	mplsLdpEntityAtmMergeCap	Syntax changed—Added enumerations vpMerge and vpAndVcMerge.
mplsLdpEntityDefaultControlVpi	mplsLdpEntityAtmDefaultControlVpi	Object name changed.
mplsLdpEntityDefaultControlVci	mplsLdpEntityAtmDefaultControlVci	Object name changed.
mplsLdpEntityUnlabTrafVpi	mplsLdpEntityAtmUnlabTrafVpi	Object name changed.
mplsLdpEntityUnlabTrafVci	mplsLdpEntityAtmUnlabTrafVci	Object name changed.
mplsLdpEntityAtmStorType	mplsLdpEntityAtmStorageType	Object name changed.
mplsLdpEntityAtmRowStatus	mplsLdpEntityAtmRowStatus	Description changed—Clarified different row status operations. Added a recommendation that the mplsLdpEntityAdminStatus object be set to down before the LSR changes the objects in this table.

MPLS-LDP-MIB Object	MPLS-LDP-ATM-STD-MIB Object	Difference
mplsLdpEntityConfAtmLRTable	mplsLdpEntityAtmLRTable	Object name changed.
mplsLdpEntityConfAtmLREntry	mplsLdpEntityAtmLREntry	Object name changed.
mplsLdpEntityConfAtmLRMinVpi	mplsLdpEntityAtmLRMinVpi	Object name changed. Description changed—0 added as a valid value.
mplsLdpEntityConfAtmLRMinVci	mplsLdpEntityAtmLRMinVci	Object name changed.
mplsLdpEntityConfAtmLRMaxVpi	mplsLdpEntityAtmRMaxVpi	Object name changed.
mplsLdpEntityConfAtmLRMaxVci	mplsLdpEntityAtmLRMaxVci	Object name changed.
mplsLdpEntityConfAtmLRStorType	mplsLdpEntityAtmLRStorageType	Object name changed.
mplsLdpEntityConfAtmLRRowStatus	mplsLdpEntityAtmLRRowStatus	Object name changed.
mplsLdpAtmSesTable	mplsLdpAtmSessionTable	Object name changed.
mplsLdpSesAtmLRLowerBoundVpi	mplsLdpSessionAtmLRLowerBoundVpi	Object name changed.
mplsLdpSesAtmLRLowerBoundVci	mplsLdpSessionAtmLRLowerBoundVci	Object name changed.
mplsLdpSesAtmLRUpperBoundVpi	mplsLdpSessionAtmLRUpperBoundVpi	Object name changed.
mplsLdpSesAtmLRUpperBoundVci	mplsLdpSessionAtmLRUpperBoundVci	Object name changed.

Differences Between the MPLS-LDP-MIB and the MPLS-LDP-GENERIC-STD-MIB (RFC 3815)

Layer 2 objects for per-platform label spaces are removed from the MPLS-LDP-MIB and placed in a new MIB module in RFC 3815. The new MIB module is the MPLS-LDP-GENERIC-STD-MIB. This action provides modularity and reduces the size of the MPLS LDP MIB.

The table below shows the difference between generic label space objects in the MPLS-LDP-MIB and objects in the MPLS-GENERIC-STD-MIB.

Table 19: MPLS LDP LSP DEC Table: MPLS-LDP-MIB and MPLS-LDP-GENERIC-STD-MIB Object Differences

MPLS-LDP-MIB Object	MPLS-LDP-GENERIC-STD-MIB Object	Difference
mplsLdpEntityConfGenLRTable	mplsLdpEntityGenericLRTable	Object name changed.
mplsLdpEntityConfGenLRMinn	mplsLdpEntityGenericLRMin	Object name changed.
mplsLdpEntityConfGenLRMax	mplsLdpEntityGenericLRMax	Object name changed.

MPLS-LDP-MIB Object	MPLS-LDP-GENERIC-STD-MIB Object	Difference
mplsLdpEntityConfGenIfIndexOrZero	mplsLdpEntityGenericIfIndexOrZero	Object name changed.
mplsLdpEntityConfGenLRStorType	mplsLdpEntityGenericLRStorageType	Object name changed.
mplsLdpEntityConfGenLRRowStatus	mplsLdpEntityGenericRowStatus	Object name changed.
—	mplsLdpEntityGenericLabelSpace	New object—A value of perPlatform(1) indicates the label space type is per platform; a value of perInterface(2) indicates the label space type is per interface.

How to Configure SNMP for MPLS EM—MPLS LDP MIB - RFC 3815

Configuring Access to an SNMP Agent on a Host NMS Workstation

To configure access to the SNMP agent on a host NMS workstation, perform the following task.

Through the use of an SNMP agent, the MPLS LDP MIBs described in RFC 3815 provide an interface for monitoring and managing LDP.

To use SNMP to manage LDP, you need to configure access to an SNMP agent on a NMS workstation. By default, the SNMP agent for the MPLS LDP MIB is disabled. Step 2 shows you how to determine if an SNMP agent is already configured, and if you need to modify the SNMP information to monitor and manage LDP.

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **configure terminal**
4. **snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [*acl-number*]
5. **do copy running-config startup-config**
6. **exit**
7. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration to determine if an SNMP agent is already running. • If no SNMP information is displayed, continue with the next step. If any SNMP information is displayed, you can modify the information or change it as needed.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [*acl-number*] **Example:** `Router(config)# snmp-server community comaccess ro`	Configures the community access string to permit access to the SNMP protocol. • The *string* argument acts like a password and permits access to the SNMP protocol. • The **view** *view-name* keyword argument pair specifies the name of a previously defined view. The view defines the objects available to the community. • The **ro** keyword specifies read-only access. Authorized management stations are able to only retrieve MIB objects. • The **rw** keyword specifies read/write access. Authorized management stations are able to both retrieve and modify MIB objects. • The *acl-number* argument is an integer from 1 to 99 that specifies an access list of IP addresses that are allowed to use the community string to gain access to the SNMP agent.
Step 5	**do copy running-config startup-config** **Example:** `Router(config)# do copy running-config startup-config`	(Optional) Saves the modified configuration to nonvolatile memory (NVRAM) as the startup configuration file. • Use this command if you made changes to the MIB information. • The **do** command allows you to perform EXEC-level commands in configuration modes.
Step 6	**exit** **Example:** `Router(config)# exit`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 7	**show running-config** **Example:** `Router# show running-config | include snmp-server`	(Optional) Displays the configuration information currently on the router. • Use the **show running-config** command to check that the **snmp-server** command statements appear in the output.

Examples

Use the **show running-config** command to display the running configuration on the host NMS workstation and examine the output for SNMP information. For example:

```
Router# show running-config | include snmp-server
snmp-server community public RO
snmp-server community private RW
```

The presence of any **snmp-server** command statement in the output that takes the form shown in this example verifies that access to the SNMP agent is configured on the host NMS workstation.

Configuring the Router to Send SNMP Notifications to a Host for Monitoring LDP

To configure the router to send SNMP notifications to a host to monitor LDP, perform the following task. The ability to display SNMP notifications helps in managing LDP sessions. You can determine if an LDP session between peers is up or down, if the path vector limits are the same for both LDP peers, and if the path vector limit threshold between the peers has been exceeded.

The **snmp-server host** command specifies which hosts receive notifications or traps. The **snmp-server enable traps** command globally enables the trap production mechanism for the specified traps.

For a host to receive a trap, an **snmp-server host** command must be configured for that host, and, generally, the trap must be enabled globally through the **snmp-server enable traps** command.

Before You Begin

Although you can set the *community-string* argument using the **snmp-server host** command by itself, we recommend that you define this string using the **snmp-server community** command prior to using the **snmp-server host** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-address* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] [**vrf** *vrf-name*]
4. **snmp-server enable traps mpls ldp** [**pv-limit**] [**session-down**] [**session-up**] [**threshold**]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**snmp-server host** *host-address* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] [**vrf** *vrf-name*]
	
	Example:
	`Router(config)# snmp-server host 172.20.2.160 traps comaccess mpls-ldp`
		Specifies the recipient of an SNMP notification operation.
		• The *host-address* argument specifies the name or Internet address of the host (the targeted recipient).
		• The **traps** keyword sends SNMP traps to this host. This is the default.
		• The **informs** keyword sends SNMP informs to this host.
		• The **version** keyword specifies the version of the SNMP used to send the traps. Version 3 is the most secure model, because it allows packet encryption with the **priv** keyword. If you use the **version** keyword, you must specify one of the following:
		  • **1—**SNMPv1. This option is not available with informs.
		  • **2c—**SNMPv2C.
		  • **3—**SNMPv3. The following three optional keywords can follow the **version 3** keyword: **auth**, **noauth**, **priv**.
		• The *community-string* argument is a password-like community string sent with the notification operation.
		• The **udp-port** *port* keyword argument pair names the UDP port of the host to use. The default is 162.
		• The *notification-type* argument specifies the type of notification to be sent to the host. If no type is specified, all notifications are sent.

	Command or Action	Purpose
		• The **vrf** *vrf-name* keyword argument pair specifies the VRF table that should be used to send SNMP notifications.
Step 4	**snmp-server enable traps mpls ldp** [**pv-limit**] [**session-down**] [**session-up**] [**threshold**] **Example:** `Router(config)# snmp-server enable traps mpls rfc ldp session-down`	Enables the router to send MPLS LDP-specific SNMP notifications (traps and informs) defined in RFC 3815. • The **pv-limit** keyword controls (enables or disables) path-vector (PV) limit notifications (mplsLdpPathVectorLimitMismatch). This notification is generated when the router establishes an LDP session with its adjacent peer LSR, but the two LSRs have dissimilar path-vector limits. • The **session-down** keyword controls (enables or disables) LDP session down notifications (mplsLdpSessionDown). This message is generated when an LDP session between the router and its adjacent LDP peer is terminated. • The **session-up** keyword controls (enables or disables) LDP session up notifications (mplsLdpSessionUp). This notification is generated when the router establishes an LDP session with another LDP entity (an adjacent LDP peer in the network). • The **threshold**keyword controls (enables or disables) PV limit notifications (mplsLdpFailedInitSessionThresholdExceeded). This notification is generated after eight failed attempts to establish an LDP session between the router and an LDP peer. The failure can be the result of any type of incompatibility between the devices.
Step 5	**end** **Example:** `Router(config)# end`	(Optional) Exits to privileged EXEC mode.

Configuring a VPN-Aware LDP MIB

Configuring SNMP Support for a VPN

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]
4. **snmp-server engineID remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engineid-string*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] **Example:** `Router(config)# snmp-server host example.com vrf trap-vrf`	Specifies the recipient of an SNMP notification operation and specifies the VRF instance table to be used for the sending of SNMP notifications. • The *host-address* argument specifies the name or Internet address of the host (the targeted recipient). • The **vrf** *vrf-name* keyword argument pair specifies the VRF table that should be used to send SNMP notifications. • The *vrf-name***traps** keyword sends SNMP traps to this host. This is the default. • The **informs** keyword sends SNMP informs to this host. • The **version** keyword specifies the version of the SNMP used to send the traps. Version 3 is the most secure model, because it allows packet encryption with the **priv** keyword. If you use the **version** keyword, you must specify one of the following: ◦ **1 —**SNMPv1. This option is not available with informs.

	Command or Action	Purpose
		◦ **2c** —SNMPv2C. ◦ **3** —SNMPv3. The following three optional keywords can follow the **version 3**keyword: **auth**, **noauth**, **priv**. • The *community-string* argument is a password-like community string sent with the notification operation. • The **udp-port** *port* keyword argument pair names the UDP port of the host to use. The default is 162. • The *notification-type* argument specifies the type of notification to be sent to the host. If no type is specified, all notifications are sent.
Step 4	**snmp-server engineID remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engineid-string* **Example:** `Router(config)# snmp-server engineID remote 172.16.20.3 vrf traps-vrf 80000009030000B064EFE100`	Specifies the SNMP engine ID of a remote SNMP device. • The *ipv4-address* argument is the IPv4 address of the device that contains the remote copy of SNMP. • The *ipv6-address* argument is the IPv6 address of the device that contains the remote copy of SNMP. • The **udp-port** keyword specifies a User Datagram Protocol (UDP) port of the host to use. • The *udp-port-number* argument is the socket number on the remote device that contains the remote copy of SNMP. The default is 161. • The **vrf** keyword specifies an instance of a routing table. • The *vrf-name* argument is the name of the VRF table to use for storing data. • The *engineid-string* is a string of a maximum of 24 characters that identifies the engine ID.
Step 5	**end** **Example:** `Router(config)# end`	Exits to privileged EXEC mode.

What to Do Next

Proceed to the Configuring an SNMP Context for a VPN, on page 40.

Configuring an SNMP Context for a VPN

To configure an SNMP context for a VPN, perform the following task. This sets up a unique SNMP context for a VPN, which allows you to access the VPN's LDP session information.

SNMP Context

SNMP contexts provide VPN users with a secure way of accessing MIB data. When a VPN is associated with a context, that VPN's specific MIB data exists in that context. Associating a VPN with a context enables service providers to manage networks with multiple VPNs. Creating and associating a context with a VPN enables a provider to prevent the users of one VPN from accessing information about users of other VPNs on the same networking device.

VPN Route Distinguishers

A route distinguisher (RD) creates routing and forwarding tables for a VPN. Cisco IOS software adds the RD to the beginning of the customer's IPv4 prefixes to change them into globally unique VPN-IPv4 prefixes.

Either the RD is an autonomous system number (ASN)-relative RD, in which case it is composed of an autonomous system number and an arbitrary number, or it is an IP-address-relative RD, in which case it is composed of an IP address and an arbitrary number. You can enter an RD in either of these formats:

- 16-bit ASN: your 32-bit number, for example, 101:3.
- 32-bit IP address: your 16-bit number, for example, 192.168.122.15:1.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server context** *context-name*
4. **ip vrf** *vrf-name*
5. **rd** *route-distinguisher*
6. **context** *context-name*
7. **route-target** {**import** | **export** | **both**} *route-target-ext-community*
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server context** *context-name*	Creates and names an SNMP context.

	Command or Action	Purpose
	Example: `Router(config)# snmp-server context context1`	• The *context-name* argument is the name of the SNMP context being created.
Step 4	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf vrf1`	Configures a VRF table and enters VRF configuration mode. • The *vrf-name* argument is the name assigned to a VRF.

Step 5	**rd** *route-distinguisher*
	Example:
	`Router(config-vrf)# rd 100:120`
		Creates a VPN route distinguisher.
		• The *route-distinguisher* argument specifies to add an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter a route distinguisher in either of these formats:
		  • 16-bit autonomous system number: your 32-bit number For example, 101:3.
		  • 32-bit IP address: your 16-bit number For example, 192.168.122.15:1.

Step 6	**context** *context-name* **Example:** `Router(config-vrf)# context context1`	Associates an SNMP context with a particular VRF. • The *context-name* argument is the name of the SNMP VPN context, up to 32 characters.
Step 7	**route-target** {**import** | **export** | **both**} *route-target-ext-community* **Example:** `Router(config-vrf)# route-target export 100:1000`	(Optional) Creates a route-target extended community for a VRF. • The **import** keyword specifies to import routing information from the target VPN extended community. • The **export** keyword specifies to export routing information to the target VPN extended community. • The **both** keyword specifies to import both import and export routing information to the target VPN extended community. • The *route-target-ext-community* argument adds the route-target extended community attributes to the VRF's list of import, export, or both (import and export) route-target extended communities.
Step 8	**end** **Example:** `Router(config-vrf)# end`	Exits to privileged EXEC mode.

What to Do Next

Proceed to the Configuring a VPN-Aware SNMP Context for SNMPv1 or SNMPv2, on page 43.

Configuring a VPN-Aware SNMP Context for SNMPv1 or SNMPv2

To configure a VPN-aware SNMP context for SNMPv1 or SNMPv2, perform the following task.

This allows you to access LDP session information for a VPN using SNMPv1 or SNMPv2.

SNMPv1 or SNMPv2 Security

SNMPv1 and SNMPv2 are not as secure as SNMPv3. SNMP Versions 1 and 2 use plain text communities and do not perform the authentication or security checks that SNMP Version 3 performs.

To configure the VPN Aware LDP MIB feature when using SNMP Version 1 or SNMP Version 2, you need to associate a community name with a VPN. This association causes SNMP to process requests coming in for a particular community string only if they come in from the configured VRF. If the community string contained in the incoming packet does not have an associated VRF, the packet is processed only if it came in through a non-VRF interface. This process prevents users outside the VPN from using a clear text community string to query the VPN data. However, this is not as secure as using SNMPv3.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server user** *username group-name* [**remote** *host* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*]
4. **snmp-server group** *group-name* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**context** *context-name*] [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*]
5. **snmp-server view** *view-name oid-tree* {**included** | **excluded**}
6. **snmp-server enable traps** [*notification-type*]
7. **snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [**notification-type**]
8. **snmp mib community-map** *community-name* [**context** *context-name*] [**engineid** *engine-id*] [**security-name** *security-name*] **target-list** *vpn-list-name*
9. **snmp mib target-list** *vpn-list-name* {**vrf** *vrf-name* | **host** *ip-address*}
10. **no snmp-server trap authentication vrf**
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server user** *username group-name* [**remote** *host* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*] **Example:** `Router(config)# snmp-server user customer1 group1 v1`	Configures a new user to an SNMP group. • The *username* argument is the name of the user on the host that connects to the agent. • The *group-name* argument is the name of the group to which the user belongs. • The **remote** *host* keyword and argument specify a remote SNMP entity to which the user belongs, and the hostname or IPv6 address or IPv4 IP address of that entity. If both an IPv6 address and IPv4 IP address are being specified, the IPv6 host must be listed first. • The **udp-port** *port* keyword and argument specify the UDP port number of the remote host. The default is UDP port 162. • The **v1** keyword specifies that SNMPv1 should be used. • The **v2c** keyword specifies that SNMPv2c should be used. • The **v3** keyword specifies that the SNMPv3 security model should be used. Allows the use of the **encrypted** and or **auth** keywords. • The **encrypted** keyword specifies whether the password appears in encrypted format (a series of digits, masking the true characters of the string). • The **auth** keyword specifies which authentication level should be used. • The **md5** keyword is the HMAC-MD5-96 authentication level. • The **sha** keyword is the HMAC-SHA-96 authentication level. • The *auth-password* argument is a string (not to exceed 64 characters) that enables the agent to receive packets from the host. The minimum length for a password is one character. The recommended length of a password is at least eight characters, and should include both letters and numbers. • The **access** *access-list* keyword and argument specify an access list to be associated with this SNMP user.
Step 4	**snmp-server group** *group-name* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**context** *context-name*] [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*]	Configures a new SNMP group or a table that maps SNMP users to SNMP views. • The *group-name* argument is the name of the group. • The **v1** keyword specifies that SNMPv1 should be used for the group.

	Command or Action	Purpose
	Example: `Router(config)# snmp-server group group1 v1 context context1 read view1 write view1 notify view1`	• The **v2c** keyword specifies that SNMPv2c should be used for the group. The SNMPv2c security model allows for the transmission of informs, and supports 64-character strings (instead of 32-character strings). • The **v3** keyword specifies that the SNMPv3 should be used for the group. SMNPv3 is the most secure of the supported security models, because it allows you to explicitly configure the authentication characteristics. • The **auth** keyword specifies authentication of a packet without encrypting it. • The **noauth** keyword specifies no authentication of a packet. • The **priv** keyword specifies authentication of a packet with encryption. • The **context** *context-name* keyword and argument associate the specified SNMP group with a configured SNMP context. • The **read** *readview* keyword and argument specify a read view for the SNMP group. The *readview* argument represents a string (not to exceed 64 characters) that is the name of the view that enables you to display only the contents of the agent. • The **write** *writeview* keyword and argument specify a write view for the SNMP group. The *writeview* argument represents a string (not to exceed 64 characters) that is the name of the view that enables you to enter data and configure the contents of the agent. • The **notify** and *notifyview* keyword argument specify a notify view for the SNMP group. The *writeview* argument represents a string (not to exceed 64 characters) that is the name of the view that enables you to specify a notify, inform, or trap. • The **access** *access-list* keyword and argument specify a standard access list (a standard ACL) to associate with the group.
Step 5	**snmp-server view** *view-name oid-tree* {**included** | **excluded**} **Example:** `Router(config)# snmp-server view view1 ipForward included`	Creates or updates a view entry. • The *view-name* argument is the label for the view record that you are updating or creating. The name is used to reference the record. • The *oid-tree* argument is the object identifier of the ASN.1 subtree to be included or excluded from the view. To identify the subtree, specify a text string consisting of numbers, such as 1.3.6.2.4, or a word, such as system. Replace a single subidentifier with the asterisk (*) wildcard to specify a subtree family; for example 1.3.*.4. • The **included** keyword configures the OID (and subtree OIDs) specified in the *oid-tree* argument to be included in the SNMP view. • The **excluded** keyword configures the OID (and subtree OIDs) specified in the *oid-tree* argument to be explicitly excluded from the SNMP view.

	Command or Action	Purpose
Step 6	**snmp-server enable traps** [*notification-type*] **Example:** `Router(config)# snmp-server enable traps`	Enables all SNMP notifications (traps or informs) available on your system. • The *notification-type* argument specifies the particular type of SNMP notification(s) to be enabled on the LSR. If a notification type is not specified, all SNMP notifications applicable to the LSR are enabled and sent to the SNMP host.

Step 7	**snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [**notification-type**]
	
	Example:
	
	`Router(config)# snmp-server host 10.0.0.1 vrf customer1 public udp-port 7002`
		Specifies the recipient of an SNMP notification operation.
		
		• The *host-address* argument specifies the name or Internet address of the host (the targeted recipient).
		
		• The **vrf** *vrf-name* keyword argument pair specifies the VRF table that should be used to send SNMP notifications.
		
		• The **traps** keyword sends SNMP traps to this host. This is the default.
		
		• The **informs** keyword sends SNMP informs to this host.
		
		• The **version** keyword specifies the version of the SNMP used to send the traps. Version 3 is the most secure model, because it allows packet encryption with the **priv** keyword. If you use the **version** keyword, you must specify one of the following:
		
		  • **1—**SNMPv1. This option is not available with informs.
		
		  • **2c —**SNMPv2C.
		
		  • **3—**SNMPv3. The following three optional keywords can follow the **version 3**keyword: **auth**, **noauth**, **priv**.
		
		• The *community-string* argument is a password-like community string sent with the notification operation.
		
		• The **udp-port** *port* keyword argument pair names the UDP port of the host to use. The default is 162.
		
		• The *notification-type* argument specifies the type of notification to be sent to the host. If no type is specified, all notifications are sent.

Step 8	**snmp mib community-map** *community-name* [**context** *context-name*] [**engineid** *engine-id*] [**security-name** *security-name*] **target-list** *vpn-list-name* **Example:** `Router(config)# snmp mib community-map community1 context context1 target-list commAVpn`	Associates an SNMP community with an SNMP context, engine ID, or security name. • The *community-name* argument is an SNMP community string. • The **context** *context-name* keyword and argument specify an SNMP context name to be mapped to the SNMP community. • The **engineid** *engine-id* keyword and argument specify an SNMP engine ID to be mapped to the SNMP community. • The **security-name** *security-name* keyword and argument specify the security name to be mapped to the SNMP community.

	Command or Action	Purpose
		• The **target-list** *vpn-list-name* keyword and argument specify the VRF list to be mapped to the SNMP community. The list name should correspond to a list name used in the **snmp mib target-list** command.
Step 9	**snmp mib target-list** *vpn-list-name* {**vrf** *vrf-name* | **host** *ip-address*} **Example:** `Router(config)# snmp mib target list commAVpn vrf vrf1`	Creates a list of target VRFs and hosts to associate with an SNMP community. • The *vpn-list-name* argument is the name of the target list. • The **vrf** keyword adds a specified VRF to the target list. • The *vrf-name*argument is the name of a VRF to include in the list. • The **host** keyword adds a specified host to the target list. • The *ip-address* argument is the IP address of the host.
Step 10	**no snmp-server trap authentication vrf** **Example:** `Router(config)# no snmp-server trap authentication vrf`	(Optional) Disables all SNMP authentication notifications (traps and informs) generated for packets received on VRF interfaces. • Use this command to disable authentication traps only for those packets on VRF interfaces with incorrect community associations.
Step 11	**end** **Example:** `Router(config) end`	Exits to privileged EXEC mode.

Configuration Examples for MPLS EM—MPLS LDP MIB - RFC 3815

Configuring Access to an SNMP Agent on a Host NMS Workstation Example

The following example shows how to configure access to an SNMP agent on a host NMS workstation:

```
configure terminal
!
snmp-server community
end
```

The following example shows how to configure access to SNMPv1 and SNMPv2C on the host NMS workstation. The configuration permits any SNMP agent to access all MPLS LDP MIB objects with read-only permission using the community string public.

```
configure terminal
```

```
!
snmp-server community public
end
```

The following example shows how to allow read-only access to all MPLS LDP MIB objects relating to members of access list 4 that specify the comaccess community string. No other SNMP agents will have access to any of the MPLS LDP MIB objects.

```
configure terminal
!
snmp-server community comaccess ro 4
end
```

Configuring the Router to Send SNMP Notifications to a Host for Monitoring LDP Example

The following example shows how to configure the router to send SNMP notifications to a host for monitoring LDP:

```
config terminal
!
snmp-server host 172.20.2.160 traps comaccess mpls-ldp
snmp-server enable traps mpls rfc ldp session-up
!
snmp-server enable traps mpls rfc ldp session-down
end
```

The session up and session down LDP notifications are configured.

Configuring a VPN-Aware LDP MIB Example

Configuring SNMP Support for a VPN Example

The following example shows how to configure SNMP support for a VPN:

```
configure terminal
!
snmp-server host 10.10.10.1 vrf traps-vrf
snmp-server engineID remote 172.16.20.3 vrf traps-vrf 80000009030000B064EFE100
end
```

Configuring an SNMP Context for a VPN Example

The following example shows how to configure an SNMP context for a VPN. In this example, the VPN vrf1 is associated with the SNMP context context1.

```
configure terminal
!
snmp-server context context1
ip vrf vrf1
 rd 100:120
 context context1
 route-target export 100:1000
 end
```

Configuring a VPN-Aware SNMP Context for SNMPv1 or SNMPv2 Example

The following configuration example shows how to configure a VPN-aware SNMP context for the MPLS LDP MIB with SNMPv1 or SNMPv2:

```
snmp-server context A
snmp-server context B
ip vrf CustomerA
 rd 100:110
 context A
 route-target export 100:1000
 route-target import 100:1000
!
ip vrf CustomerB
 rd 100:120
 context B
 route-target export 100:2000
 route-target import 100:2000
!
interface Ethernet3/1
 description Belongs to VPN A
 ip vrf forwarding CustomerA
 ip address 10.0.0.0 255.255.0.0

interface Ethernet3/2
 description Belongs to VPN B
 ip vrf forwarding CustomerB
 ip address 10.0.0.1 255.255.0.0
snmp-server user commA grp1A v1
snmp-server user commA grp2A v2c
snmp-server user commB grp1B v1
snmp-server user commB grp2B v2c
snmp-server group grp1A v1 context A read viewA write viewA notify viewA
snmp-server group grp1B v1 context B read viewB write viewB notify viewB
snmp-server view viewA ipForward included
snmp-server view viewA ciscoPingMIB included
snmp-server view viewB ipForward included
snmp-server view viewB ciscoPingMIB included
snmp-server enable traps
snmp-server host 10.0.0.3 vrf CustomerA commA udp-port 7002
snmp-server host 10.0.0.4 vrf CustomerB commB udp-port 7002
snmp mib community-map commA context A target-list commAvpn
! Configures source address validation
snmp mib community-map commB context B target-list commBvpn
! Configures source address validation
snmp mib target list commAvpn vrf CustomerA
! Configures a list of VRFs or from which community commA is valid
snmp mib target list commBvpn vrf CustomerB
! Configures a list of VRFs or from which community commB is valid
```

Additional References

Related Documents

Related Topic	Document Title
MPLS LDP concepts and configuration tasks	MPLS LDP Configuration Guide
A description of SNMP agent support in the Cisco IOS software for the MPLS Label Switching Router MIB (MPLS-LSR-STD-MIB)	MPLS EM—MPLS LSR MIB - RFC 3813

Related Topic	Document Title
SNMP commands	Network Management Command Reference
SNMP configuration	"Configuring SNMP Support" chapter in the Network Management Configuration Guide
SNMP support for VPNs	SNMP Notification Support for VPNs
SNMP context support for VPNs configuration tasks	SNMP Support over VPNs—Context Based Access Control
MPLS concepts and configuration tasks	Basic MPLS Configuration Guide
CEF concepts and configuration tasks	"Configuring Cisco Express Forwarding" section in the IP Switching Configuration Guide
Information about MPLS EM	Cisco IOS MPLS Embedded Management Application Note Cisco IOS MPLS Embedded Management Q&A

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIB	MIBs Link
• MPLS-LDP-STD-MIB • MPLS-LDP-ATM-STD-MIB • MPLS-LDP-FRAME-RELAY-STD-MIB • MPLS-LDP-GENERIC-STD-MIB	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 3036	LDP Specification

RFC	Title
RFC 3037	LDP Applicability
RFC 3815	*Definitions of Managed Objects for the Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP)*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for MPLS EM—MPLS LDP MIB - RFC 3815

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 20: Feature Information for MPLS EM—MPLS LDP MIB - RFC 3815

Feature Name	Releases	Feature Information
MPLS EM—MPLS LDP MIB - RFC 3815	12.2(33)SRB 12.2(33)SB	

Feature Name	Releases	Feature Information
		The MPLS EM—MPLS LDP MIB - RFC 3815 feature document describes the MIBs that support the Multiprotocol Label Switching (MPLS) Label Distribution Protocol (LDP) based on RFC 3815, *Definitions of Managed Objects for the Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP)* , and describes the differences between RFC 3815 and the MPLS-LDP-MIB based on the Internet Engineering Task Force (IETF) draft Version 8 (draft-ieft-mpls-ldp-08.txt). RFC 3815 and IETF draft Version 8 provide an interface for managing LDP through the use of the Simple Network Management Protocol (SNMP). In RFC 3815, the content of the MPLS-LDP-MIB is divided into four MIB modules: the MPLS-LDP-STD-MIB, the MPLS-LDP-GENERIC-STD-MIB, the MPLS-LDP-ATM-STD-MIB, and the MPLS-LDP-FRAME-RELAY-STD-MIB. Cisco IOS MPLS Embedded Management (EM) is a set of standards and value-added services that facilitate the deployment, operation, administration, and management of MPLS-based networks in line with the fault, configuration, accounting, performance, and security (FCAPS) model. In 12.2(33)SRB, this feature was introduced. In 12.2(33)SB, the feature was integrated into Cisco IOS Release 12.2SB . The following command was introduced in this feature: **snmp-server enable traps mpls**

Feature Name	Releases	Feature Information
		rfc ldp.

Glossary

FPI —forwarding path identifier. An identifier required to locate Multiprotocol Label Switching (MPLS) forwarding information for a forwarding equivalence class (FEC). Examples of types of FPIs supported by the MPLS Forwarding Infrastructure (MFI) are IPv4, IPv6, LABEL, SSS, and TE.

LDP —Label Distribution Protocol. A standard protocol between MPLS-enabled routers that is used for the negotiation of the labels (addresses) used to forward packets.

LFIB —Label Forwarding Information Base. A data structure and way of managing forwarding in which destinations and incoming labels are associated with outgoing interfaces and labels.

LSP —label switched path. A sequence of hops in which a packet travels from one router to another router by means of label switching mechanisms. A label switched path can be established dynamically, based on normal routing mechanisms, or through configuration.

LSR —label switch router. A device that forwards MPLS packets based on the value of a fixed-length label encapsulated in each packet.

MFI —MPLS Forwarding Infrastructure. In the Cisco MPLS subsystem, the data structure for storing information about incoming and outgoing labels and associated equivalent packets suitable for labeling.

MIB —Management Information Base. Database of network management information that is used and maintained by a network management protocol such as SNMP. The value of a MIB object can be changed or retrieved by means of SNMP commands, usually through a network management system. MIB objects are organized in a tree structure that includes public (standard) and private (proprietary) branches.

MOI —MPLS output information. The MOI includes the next hop, outgoing interface, and outgoing label.

MPLS —Multiprotocol Label Switching. MPLS is a method for forwarding packets (frames) through a network. It enables routers at the edge of a network to apply labels to packets (frames). ATM switches or existing routers in the network core can switch packets according to the labels with minimal lookup overhead.

NMS —network management station. A device (usually a workstation) that performs Simple Network Management Protocol (SNMP) queries to the SNMP agent of a managed device to retrieve or modify information.

notification request —Message sent by an SNMP agent to a network management station, console, or terminal, indicating that a significant event occurred. SNMP notification requests are more reliable than traps, because a notification request from an SNMP agent requires that the SNMP manager acknowledge receipt of the notification request. The manager replies with an SNMP response protocol data unit (PDU). If the manager does not receive a notification message from an SNMP agent, it does not send a response. If the sender (SNMP agent) never receives a response, the notification request can be sent again. Thus, a notification request is more likely than a trap to reach its intended destination.

SNMP —Simple Network Management Protocol. Management protocol used almost exclusively in TCP/IP networks. SNMP provides a means for monitoring and controlling network devices, and for managing configurations, statistics collection, performance, and security.

trap —Message sent by an SNMP agent to a network management station, console, or terminal, indicating that a significant event occurred. Traps are less reliable than notification requests, because the receiver does not send an acknowledgment when it receives a trap. The sender cannot determine if the trap was received.