MPLS: Embedded Management and MIBs Configuration Guide, Cisco IOS Release 15SY

First Published: October 15, 2012

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

MPLS LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

The MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV feature helps service providers monitor label switched paths (LSPs) and quickly isolate Multiprotocol Label Switching (MPLS) forwarding problems.

The feature provides the following capabilities:

- MPLS LSP ping to test LSP connectivity for IPv4 Label Distribution Protocol (LDP) prefixes, Resource Reservation Protocol (RSVP) traffic engineering (TE), and Any Transport over MPLS (AToM) forwarding equivalence classes (FECs).
- MPLS LSP traceroute to trace the LSPs for IPv4 LDP prefixes and RSVP TE prefixes.

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 LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

Before you use the MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV feature, you should:

- Determine the baseline behavior of your MPLS network. For example:
    - Expected MPLS experimental (EXP) treatment.
    - Expected maximum size packet or maximum transmission unit (MTU) of the LSP.
    - The topology, expected label switched path, and number of links in the LSP. Trace the paths of the label switched packets including the paths for load balancing.
- Understand how to use MPLS and MPLS applications. You need to:
    - Know how LDP is configured.
    - Understand AToM concepts.
- Understand label switching, forwarding, and load balancing.

Before using the **ping mpls** or **trace mpls** command, you must ensure that the router is configured to encode and decode MPLS echo packets in a format that all receiving routers in the network can understand.

Restrictions for MPLS LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

- You cannot use MPLS LSP traceroute to trace the path taken by AToM packets. MPLS LSP traceroute is not supported for AToM. (MPLS LSP ping is supported for AToM.) However, you can use MPLS LSP traceroute to troubleshoot the Interior Gateway Protocol (IGP) LSP that is used by AToM.
- You cannot use MPLS LSP ping to validate or trace MPLS Virtual Private Networks (VPNs).
- You cannot use MPLS LSP traceroute to troubleshoot LSPs that employ time-to-live (TTL) hiding.
- MPLS supports per-destination and per-packet (round robin) load balancing. If per-packet load balancing is in effect, you should not use MPLS LSP traceroute because LSP traceroute at a transit router consistency checks the information supplied in the previous echo response from the directly connected upstream router. When round robin is employed, the path that an echo request packet takes cannot be controlled in a way that allows a packet to be directed to TTL expire at a given router. Without that ability, the consistency checking may fail during an LSP traceroute. A consistency check failure return code may be returned.
- A platform must support LSP ping and traceroute in order to respond to an MPLS echo request packet.
- Unless the MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV feature is enabled along the entire path, you cannot get a reply if the request fails along the path at any node.

- There are certain limitations when a mixture of draft versions are implemented within a network. The version of the draft must be compatible with Cisco's implementation. Due to the way the LSP Ping draft was written, earlier versions may not be compatible with later versions because of changes to type, length, values (TLVs) formats without sufficient versioning information. Cisco attempts to compensate for this in its implementations by allowing the sending and responding routers to be configured to encode and decode echo packets assuming a certain version.
- If you want to use MPLS LSP traceroute, the network should not use TTL hiding.

Information About MPLS LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

MPLS LSP Ping Traceroute for LDP TE and LSP Ping for VCCV Functionality

Internet Control Message Protocol (ICMP) ping and traceroute are often used to help diagnose the root cause when a forwarding failure occurs. However, they are not well suited for identifying LSP failures because an ICMP packet can be forwarded via IP to the destination when an LSP breakage occurs.

The MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV feature is well suited for identifying LSP breakages for the following reasons:

- An MPLS echo request packet cannot be forwarded via IP because IP TTL is set to 1 and the IP destination address field is set to a 127/8 address.
- The FEC being checked is not stored in the IP destination address field (as is the case of ICMP).

MPLS echo request and reply packets test LSPs. There are two methods by which a downstream router can receive packets:

- The Cisco implementation of MPLS echo request and echo reply that was previously based on the Internet Engineering Task Force (IETF) Internet Draft Detecting MPLS Data Plane Failures (draft-ietf-mpls-lsp-ping-03.txt).
- Features described in this document that are based on the IETF RFC 4379 Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures :
    - Echo request output interface control
    - Echo request traffic pacing
    - Echo request end-of-stack explicit-null label shimming
    - Echo request request-dsmap capability
    - Request-fec checking
    - Depth limit reporting

MPLS LSP Ping Operation

MPLS LSP ping uses MPLS echo request and reply packets to validate an LSP. You can use MPLS LSP ping to validate IPv4 LDP, AToM, and IPv4 RSVP FECs by using appropriate keywords and arguments with the **ping mpls**command.

The MPLS echo request packet is sent to a target router through the use of the appropriate label stack associated with the LSP to be validated. Use of the label stack causes the packet to be forwarded over the LSP itself.

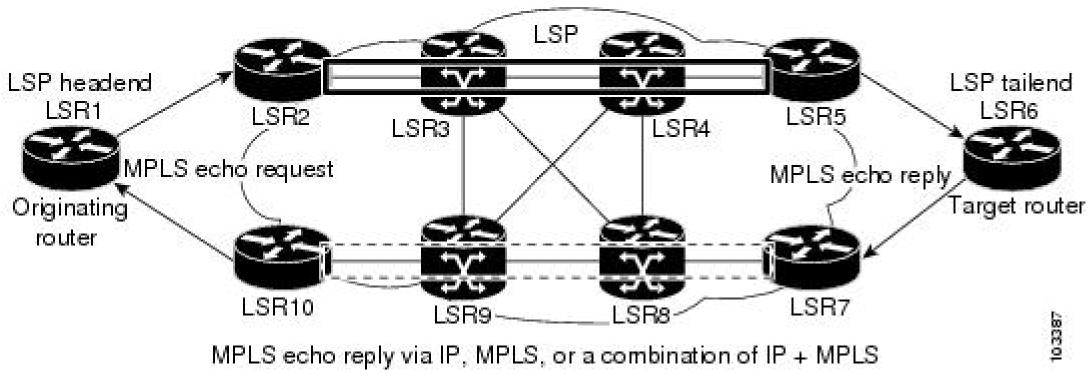
The destination IP address of the MPLS echo request packet is different from the address used to select the label stack. The destination IP address is defined as a 127.*x* .*y* .*z* /8 address. The 127.*x* .*y* .*z* /8 address prevents the IP packet from being IP switched to its destination if the LSP is broken.

An MPLS echo reply is sent in response to an MPLS echo request. The reply is sent as an IP packet and it is forwarded using IP, MPLS, or a combination of both types of switching. The source address of the MPLS echo reply packet is an address obtained from the router generating the echo reply. The destination address is the source address of the router that originated the MPLS echo request packet.

The MPLS echo reply destination port is set to the echo request source port.

The figure below shows MPLS LSP ping echo request and echo reply paths.

Figure 1: MPLS LSP Ping Echo Request and Echo Reply Paths



If you initiate an MPLS LSP ping request at LSR1 to a FEC at LSR6, you get the results shown in the table below.

Table 1: MPLS LSP Ping Example

Step	Router	Action
1.	LSR1	Initiates an MPLS LSP ping request for an FEC at the target router LSR6 and sends an MPLS echo request to LSR2.
2.	LSR2	Receives the MPLS echo request packet and forwards it through transit routers LSR3 and LSR4 to the penultimate router LSR5.

Step	Router	Action
3.	LSR5	Receives the MPLS echo request, pops the MPLS label, and forwards the packet to LSR6 as an IP packet.
4.	LSR6	Receives the IP packet, processes the MPLS echo request, and sends an MPLS echo reply to LSR1 through an alternate route.
5.	LSR7 to LSR10	Receives the MPLS echo reply and forwards it back toward LSR1, the originating router.
6.	LSR1	Receives the MPLS echo reply in response to its MPLS echo request.

MPLS LSP Traceroute Operation

MPLS LSP traceroute uses MPLS echo request and reply packets to validate an LSP. You can use MPLS LSP traceroute to validate IPv4 LDP and IPv4 RSVP FECs by using appropriate keywords and arguments with the **trace mpls** command.

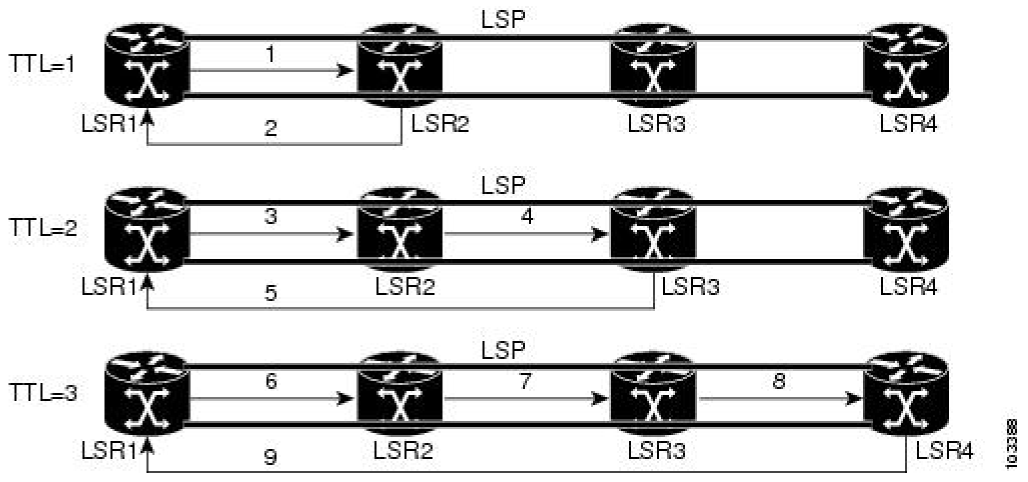
The MPLS LSP Traceroute feature uses TTL settings to force expiration of the TTL along an LSP. MPLS LSP Traceroute incrementally increases the TTL value in its MPLS echo requests (TTL = 1, 2, 3, 4) to discover the downstream mapping of each successive hop. The success of the LSP traceroute depends on the transit router processing the MPLS echo request when it receives a labeled packet with a TTL = 1. On Cisco routers, when the TTL expires, the packet is sent to the Route Processor (RP) for processing. The transit router returns an MPLS echo reply containing information about the transit hop in response to the TTL-expired MPLS packet.

The MPLS echo reply destination port is set to the echo request source port.

Note When a router traces an IPV4 FEC that goes over a traffic engineering tunnel, intermediate routers may return U (unreachable) if LDP is not running in those intermediate routers.

The figure below shows an MPLS LSP traceroute example with an LSP from LSR1 to LSR4.

Figure 2: MPLS LSP Traceroute Example



If you enter an LSP traceroute to an FEC at LSR4 from LSR1, you get the results shown in the table below.

Table 2: MPLS LSP Traceroute Example

Step	Router	MPLS Packet Type and Description	Router Action (Receive or Send)
1.	LSR1	MPLS echo request--With a target FEC pointing to LSR4 and to a downstream mapping	• Sets the TTL of the label stack to 1 • Sends the request to LSR2
2.	LSR2	MPLS echo reply	• Receives the packet with a TTL = 1 • Processes the User Datagram Protocol (UDP) packet as an MPLS echo request • Finds a downstream mapping and replies to LSR1 with its own downstream mapping, based on the incoming label
3.	LSR1	MPLS echo request--With the same target FEC and the downstream mapping received in the echo reply from LSR2	• Sets the TTL of the label stack to 2 • Sends the request to LSR2

Step	Router	MPLS Packet Type and Description	Router Action (Receive or Send)
4.	LSR2	MPLS echo request	• Receives the packet with a TTL = 2 • Decrements the TTL • Forwards the echo request to LSR3
5.	LSR3	MPLS reply packet	• Receives the packet with a TTL = 1 • Processes the UDP packet as an MPLS echo request • Finds a downstream mapping and replies to LSR1 with its own downstream mapping based on the incoming label
6.	LSR1	MPLS echo request--With the same target FEC and the downstream mapping received in the echo reply from LSR3	• Sets the TTL of the packet to 3 • Sends the request to LSR2
7.	LSR2	MPLS echo request	• Receives the packet with a TTL = 3 • Decrements the TTL • Forwards the echo request to LSR3
8.	LSR3	MPLS echo request	• Receives the packet with a TTL = 2 • Decrements the TTL • Forwards the echo request to LSR4

Step	Router	MPLS Packet Type and Description	Router Action (Receive or Send)
9.	LSR4	MPLS echo reply	• Receives the packet with a TTL = 1 • Processes the UDP packet as an MPLS echo request • Finds a downstream mapping and also finds that the router is the egress router for the target FEC • Replies to LSR1

MPLS Network Management with MPLS LSP Ping and MPLS LSP Traceroute

To manage an MPLS network, you must have the ability to monitor LSPs and quickly isolate MPLS forwarding problems. You need ways to characterize the liveliness of an LSP and reliably detect when an LSP fails to deliver user traffic.

You can use MPLS LSP ping to verify the LSP that is used to transport packets destined for IPv4 LDP prefixes, and AToM PW FECs. You can use MPLS LSP traceroute to trace LSPs that are used to carry packets destined for IPv4 LDP prefixes.

An MPLS echo request is sent through an LSP to validate it. A TTL expiration or LSP breakage causes the transit router to process the echo request before it gets to the intended destination. The router returns an MPLS echo reply that contains an explanatory reply code to the originator of the echo request.

The successful echo request is processed at the egress of the LSP. The echo reply is sent via an IP path, an MPLS path, or a combination of both back to the originator of the echo request.

Any Transport over MPLS Virtual Circuit Connection

AToM Virtual Circuit Connection Verification (VCCV) allows you to send control packets inband of an AToM PW from the originating provider edge (PE) router. The transmission is intercepted at the destination PE router, instead of being forwarded to the customer edge (CE) router. This capability allows you to use MPLS LSP ping to test the PW section of AToM virtual circuits (VCs).

LSP ping allows verification of AToM VC setup by FEC 128 or FEC 129. FEC 128-based AToM VCs can be set up by using LDP for signaling or by using a static pseudowire configuration without using any signaling component on the two endpoints. Cisco software does not distinguish between FEC 128 and FEC 129 static pseudowires while issuing MPLS ping; the same commands are used.

AToM VCCV consists of the following:

- A signaled component in which the AToM VCCV capabilities are advertised during VC label signaling
- A switching component that causes the AToM VC payload to be treated as a control packet

AToM VCCV Signaling

One of the steps involved in AToM VC setup is the signaling or communication of VC labels and AToM VCCV capabilities between AToM VC endpoints. To communicate the AToM VCCV disposition capabilities of each endpoint, the router uses an optional parameter, defined in the IETF Internet Draft *Pseudo Wire (PW) Virtual Circuit Connection Verification (VCCV)* (draft-ieft-pwe3-vccv-01).

The AToM VCCV disposition capabilities are categorized as follows:

- Applications--MPLS LSP ping and ICMP ping are applications that AToM VCCV supports to send packets inband of an AToM PW for control purposes.
- Switching modes--Type 1 and Type 2 are switching modes that AToM VCCV uses for differentiating between control and data traffic.

The table below describes AToM VCCV Type 1 and Type 2 switching modes.

Table 3: Type 1 and Type 2 AToM VCCV Switching Modes

Switching Mode	Description
Type 1	Uses a Protocol ID (PID) field in the AToM control word to identify an AToM VCCV packet
Type 2	Uses an MPLS Router Alert Label above the VC label to identify an AToM VCCV packet

Selection of AToM VCCV Switching Types

Cisco routers always use Type 1 switching, if available, when they send MPLS LSP ping packets over an AToM VC control channel. Type 2 switching accommodates those VC types and implementations that do not support or interpret the AToM control word.

The table below shows the AToM VCCV switching mode advertised and the switching mode selected by the AToM VC.

Table 4: AToM VCCV Switching Mode Advertised and Selected by AToM VC

Type Advertised	Type Selected
AToM VCCV not supported	--
Type 1 AToM VCCV switching	Type 1 AToM VCCV switching
Type 2 AToM VCCV switching	Type 2 AToM VCCV switching
Type 1 and Type 2 AToM VCCV switching	Type 1 AToM VCCV switching

An AToM VC advertises its AToM VCCV disposition capabilities in both directions: that is, from the originating router (PE1) to the destination router (PE2), and from PE2 to PE1.

In some instances, AToM VCs might use different switching types if the two endpoints have different AToM VCCV capabilities. If PE1 supports Type 1 and Type 2 AToM VCCV switching and PE2 supports only Type 2 AToM VCCV switching, there are two consequences:

- LSP ping packets sent from PE1 to PE2 are encapsulated with Type 2 switching.
- LSP ping packets sent from PE2 to PE1 use Type 1 switching.

You can determine the AToM VCCV capabilities advertised to and received from the peer by entering the **show mpls l2transport binding** command at the PE router.

Information Provided by the Router Processing LSP Ping or LSP Traceroute

The table below describes the characters that the router processing an LSP ping or LSP traceroute packet returns to the sender about the failure or success of the request.

You can also display the return code for an MPLS LSP Ping operation if you enter the **ping mpls verbose** command.

Table 5: Echo Reply Return Codes

Output Code	Echo Return Code	Meaning
x	0	No return code.
M	1	Malformed echo request.
m	2	Unsupported TLVs.
!	3	Success.
F	4	No FEC mapping.
D	5	DS Map mismatch.
I	6	Unknown Upstream Interface index.
U	7	Reserved.
L	8	Labeled output interface.
B	9	Unlabeled output interface.
f	10	FEC mismatch.
N	11	No label entry.
P	12	No receive interface label protocol.
p	13	Premature termination of the LSP.

Output Code	Echo Return Code	Meaning
X	unknown	Undefined return code.

Note Echo return codes 6 and 7 are accepted only for Version 3 (draft-ieft-mpls-ping-03).

IP Does Not Forward MPLS Echo Request Packets

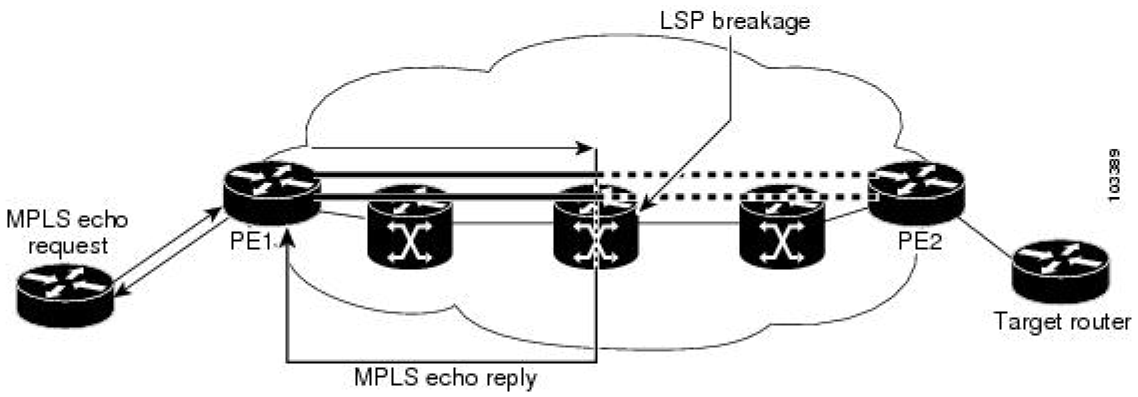
MPLS echo request packets sent during an LSP ping are never forwarded by IP. The IP header destination address field in an MPLS echo request packet is a 127.*x.y.z* /8 address. Routers should not forward packets using a 127.*x.y.z* /8 address. The 127.*x.y.z* /8 address corresponds to an address for the local host.

Use of a 127.*x* .*y* .*z* address as the destination address of the UDP packet is significant because the MPLS echo request packet fails to make it to the target router if a transit router does not label switch the LSP. The use of the 127.*x* .*y* .*z* address allows for the detection of LSP breakages. The following occurs at the transit router:

- If an LSP breakage occurs at a transit router, the MPLS echo packet is not forwarded; it is consumed by the router.
- If the LSP is intact, the MPLS echo packet reaches the target router and is processed by the terminal point of the LSP.

The figure below shows the path of the MPLS echo request and reply when a transit router fails to label switch a packet in an LSP.

Figure 3: Path when Transit Router Fails to Label Switch a Packet



Note An AToM payload does not contain usable forwarding information at a transit router because the payload may not be an IP packet. An MPLS VPN packet, although an IP packet, does not contain usable forwarding information at a transit router because the destination IP address is significant only to the virtual routing and forwarding (VRF) instances at the endpoints of the MPLS network.

Compatibility Between the MPLS LSP and Ping or Traceroute Implementations

LSP ping drafts after Version 3 (draft-ietf-mpls-ping-03) have undergone numerous TLV format changes, but the versions of the draft do not always interoperate.

To allow later Cisco implementations to interoperate with draft Version 3 Cisco and non-Cisco implementations, use a global configuration mode to decode echo packets in formats understood by draft Version 3 implementations.

Unless configured otherwise, a Cisco implementation encodes and decodes echo requests assuming the version on which the IETF implementations is based.

To prevent failures reported by the replying router due to TLV version issues, you should configure all routers in the core. Encode and decode MPLS echo packets in the same draft version. For example, if the network is running RFC 4379 (Cisco Version 4) implementations but one router is capable of only Version 3 (Cisco Revision 3), configure all routers in the network to operate in Revision 3 mode.

The Cisco implementation of MPLS echo request and echo reply is based on the IETF RFC 4379. IEFT drafts subsequent to this RFC (drafts 3, 4, 5, 6, and 7) introduced TLV format differences. These differences could not be identified because the echo packet had no way to differentiate between one TLV format and another TLV format. This introduced limited compatibility between the MPLS LSP Ping Traceroute implementations in the Cisco IOS 12.0(27)S1 and 12.0(27)S2 releases and the MPLS ping or traceroute implementation in later Cisco IOS releases. To allow interoperability between these releases, a **revision** keyword was added for the **ping mpls** and **trace mpls** commands. The **revision** keyword enables Cisco IOS releases to support the existing draft changes and any changes from future versions of the IETF LSP Ping draft.

Note We recommend that you use the **mpls oam** global configuration command instead of the revision option.

Note No images are available on cisco.com to support Revision 2. It is recommended that you use only images supporting Version 3 and later when configuring TLV encode and decode modes. MPLS Multipath LSP traceroute requires Cisco Revision 4 or later.

CiscoVendorExtensions

In Cisco's Version 3 (draft-ietf-mpls-ping-03.txt) implementations, Cisco defined a vendor extension TLV in the ignore-if-not-understood TLV space. It is used for the following purposes:

- Provide an ability to track TLV versions.
- Provide an experimental Reply TOS capability.

The first capability was defined before the existence of the global configuration command for setting the echo packet encode and decode behavior. TLV version information in an echo packet overrides the configured decoding behavior. Using this TLV for TLV versions is no longer required since the introduction of the global configuration capability.

The second capability controls the reply DSCP. Draft Version 8 defines a Reply TOS TLV, so the use of the reply DSCP is no longer required.

You enable compatibility between the MPLS LSP and ping or traceroute implementation by customizing the default behavior of echo packets.

DSCP Option to Request a Specific Class of Service in an Echo Reply

Cisco software includes a reply differentiated services code point (DSCP) option that lets you request a specific class of service (CoS) in an echo reply.

The reply DSCP option is supported in the experimental mode for IETF draft-ietf-mpls-lsp-ping-03.txt. Cisco implemented a vendor-specific extension for the reply DSCP option rather than using a Reply TOS TLV. A Reply TOS TLV serves the same purpose as the **reply dscp** command in RFC 4379. This draft provides a standardized method of controlling the reply DSCP.

Note Before draft Version 8, Cisco implemented the Reply DSCP option as an experimental capability using a Cisco vendor extension TLV. If a router is configured to encode MPLS echo packets for draft Version 3 implementations, a Cisco vendor extension TLV is used instead of the Reply TOS TLV that was defined in draft Version 8.

Reply Modes for an MPLS LSP Ping and LSP Traceroute Echo Request Response

The reply mode controls how a responding router replies to an MPLS echo request sent by a **ping mpls** or **trace mpls** command. There are two reply modes for an echo request packet:

- ipv4--Reply with an IPv4 UDP packet (default)
- router-alert--Reply with an IPv4 UDP packet with router alert

Note It is useful to use ipv4 and router-alert reply modes in conjunction with each other to prevent false negatives. If you do not receive a reply via the ipv4 mode, it is useful to send a test with the router-alert reply mode. If both fail, something is wrong in the return path. The problem may be only that the Reply TOS is not set correctly.

IPv4 Reply Mode

IPv4 packet is the most common reply mode used with a **ping mpls** or **trace mpls** command when you want to periodically poll the integrity of an LSP. With this option, you do not have explicit control over whether the packet traverses IP or MPLS hops to reach the originator of the MPLS echo request. If the originating (headend) router fails to receive a reply to an MPLS echo request when you use the **reply mode ipv4** keywords, use the **reply mode router-alert** keywords.

Router-Alert Reply Mode

The router-alert reply mode adds the router alert option to the IP header. When an IP packet that contains an IP router alert option in its IP header or an MPLS packet with a router alert label as its outermost label arrives at a router, the router punts (redirects) the packet to the Route Processor (RP) level for handling. This forces the Cisco router to handle the packet at each intermediate hop as it moves back to the destination. Hardware and line-card forwarding inconsistencies are bypassed. Router-alert reply mode is more expensive than IPv4 mode because the reply goes hop-by-hop. It also is slower, so the sender receives a reply in a relatively longer period of time.

The table below describes how IP and MPLS packets with an IP router alert option are handled by the router switching path processes.

Table 6: Path Process Handling of IP and MPLS Router Alert Packets

Incoming Packet	Normal Switching Action	Process Switching Action	Outgoing Packet
IP packet--Router alert option in IP header	Router alert option in IP header causes the packet to be punted to the process switching path.	Forwards the packet as is	IP packet--Router alert option in IP header
		Forwards the packet as is	MPLS packet
MPLS packet-- Outermost label contains a router alert	If the router alert label is the outermost label, it causes the packet to be punted to the process switching path.	Removes the outermost router alert label and forwards the packet as an IP packet	IP packet--Router alert option in IP header
		Preserves the outermost router alert label and forwards the MPLS packet	MPLS packet-- Outermost label contains a router alert.

LSP Breaks

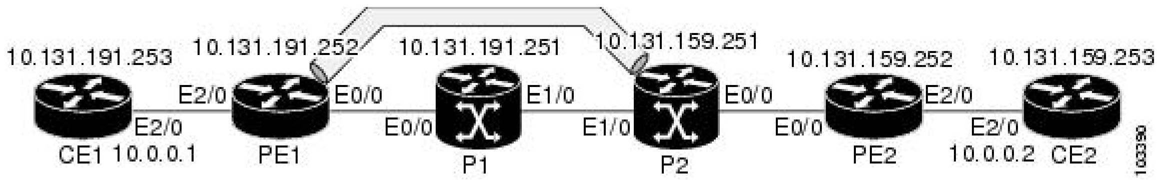
If there is a problem forwarding MPLS packets in your network, you can determine where there are LSP breaks. This section describes MTU discovery in an LSP.

Untagged output interfaces at a penultimate hop do not impact the forwarding of IP packets through an LSP because the forwarding decision is made at the penultimate hop through use of the incoming label. However, untagged output interfaces cause AToM and MPLS VPN traffic to be dropped at the penultimate hop.

During an MPLS LSP ping, MPLS echo request packets are sent with the IP packet attribute set to "do not fragment." That is, the Don't Fragment (DF) bit is set in the IP header of the packet. This allows you to use the MPLS echo request to test for the MTU that can be supported for the packet through the LSP without fragmentation.

The figure below shows a sample network with a single LSP from PE1 to PE2 formed with labels advertised by the LDP.

Figure 4: Sample Network with LSP--Labels Advertised by LDP



You can determine the maximum receive unit (MRU) at each hop by using the MPLS Traceroute feature to trace the LSP. The MRU is the maximum size of a labeled packet that can be forwarded through an LSP.

How to Configure MPLS LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

Enabling Compatibility Between the MPLS LSP and Ping or Traceroute Implementation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mplsoam**
4. **echo revision** {**3** | **4**}
5. **echo vendor-extension**
6. **exit**

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.

	Command or Action	Purpose
Step 3	**mplsoam** **Example:** `Router(config)# mpls oam`	Enters MPLS OAM configuration mode for customizing the default behavior of echo packets.

Step 4	**echo revision** {**3** | **4**}
	
	Example:
	
	`Router(config-mpls)# echo revision 4`
		Specifies the revision number of the echo packet's default values.
		• 3--draft-ietf-mpls-ping-03 (Revision 2).
		• 4--RFC 4379 compliant (default).

Step 5	**echo vendor-extension** **Example:** `Router(config-mpls)# echo vendor-extension`	Sends the Cisco-specific extension of TLVs with echo packets.
Step 6	**exit** **Example:** `Router(config-mpls)# exit`	Returns to global configuration mode.

Validating an LDP IPv4 FEC by Using MPLS LSP Ping and MPLS LSP Traceroute

SUMMARY STEPS

1. **enable**
2. Do one of the following:
   - **ping mpls ipv4** *destination-address /destination-mask-length* [**repeat** *count*] [**exp** *exp-bits*] [**verbose**]
   - **trace mpls ipv4** *destination-address /destination-mask-length*
3. **exit**

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	Do one of the following:
	• **ping mpls ipv4** *destination-address /destination-mask-length* [**repeat** *count*] [**exp** *exp-bits*] [**verbose**]
	• **trace mpls ipv4** *destination-address /destination-mask-length*
	Example:
	`Router# ping mpls ipv4 10.131.191.252/32 exp 5 repeat 5 verbose`
	Example:
	`Router# trace mpls ipv4 10.131.191.252/32`
		Selects an LDP IPv4 prefix FEC for validation.

Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Validating a Layer 2 FEC by Using MPLS LSP Ping and MPLS LSP Traceroute

SUMMARY STEPS

1. **enable**
2. **ping mpls pseudowire** *ipv4-address* **vc-id** *vc-id*
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**ping mpls pseudowire** *ipv4-address* **vc-id** *vc-id* **Example:** `Router# ping mpls pseudowire 10.131.191.252 vc-id 333`	Selects a Layer 2 FEC for validation.
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Using DSCP to Request a Specific Class of Service in an Echo Reply

SUMMARY STEPS

1. **enable**
2. Do one of the following:
    - **ping mpls** {**ipv4** *destination-address/destination-mask-length* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**reply dscp** *dscp-value*]
    - **trace mpls ipv4** *destination-address/destination-mask-length* [**reply dscp** *dscp-value*]
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Do one of the following: • **ping mpls** {**ipv4** *destination-address/destination-mask-length* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**reply dscp** *dscp-value*] • **trace mpls ipv4** *destination-address/destination-mask-length* [**reply dscp** *dscp-value*]	Controls the DSCP value of an echo reply.

	Command or Action	Purpose

	Example:
	`Router# ping mpls ipv4 10.131.191.252/32 reply dscp 50`
	Example:
	`Router# trace mpls ipv4 10.131.191.252/32 reply dscp 50`	

Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Controlling How a Responding Router Replies to an MPLS Echo Request

SUMMARY STEPS

1. **enable**
2. Do one of the following:
   - **ping mpls** {**ipv4***destination-address*/*destination-mask-length* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **reply mode** {**ipv4** | **router-alert**}
   - **trace mpls ipv4** *destination-address*/*destination-mask* **reply mode** {**ipv4** | **router-alert**}
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Do one of the following: • **ping mpls** {**ipv4***destination-address*/*destination-mask-length* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **reply mode** {**ipv4** | **router-alert**}	Checks MPLS LSP connectivity. or Discovers MPLS LSP routes that packets actually take when traveling to their destinations.

	Command or Action	Purpose

	• **trace mpls ipv4** *destination-address/destination-mask* **reply mode** {**ipv4** | **router-alert**}
	
	Example:
	`Router# ping mpls ipv4 10.131.191.252/32 reply mode ipv4`
	
	Example:
	`Router# trace mpls ipv4 10.131.191.252/32 reply mode router-alert`
		Note To specify the reply mode, you must enter the **reply mode** keyword with the **ipv4** or the **router-alert** keyword.

Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Using MPLS LSP Ping to Discover Possible Loops

With the MPLS LSP Ping feature, loops can occur if you use the UDP destination address range, repeat option, or sweep option.

To use MPLS LSP ping to discover possible loops, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **ping mpls** {**ipv4** *destination-address/destination-mask* [**destination** *address-start address-end increment* | [**pseudowire** *ipv4-address* **vc-id** *vc-id address-end increment*]} [**repeat** *count*] [**sweep** *minimum maximum size-increment*]
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**ping mpls** {**ipv4** *destination-address/destination-mask* [**destination** *address-start address-end increment* | [**pseudowire** *ipv4-address* **vc-id** *vc-id address-end increment*]} [**repeat** *count*] [**sweep** *minimum maximum size-increment*] **Example:** `Router# ping mpls ipv4 10.131.159.251/32 destination 127.0.0.1 127.0.0.2 1 repeat 2 sweep 1450 1475 25`	Checks MPLS LSP connectivity.
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Using MPLS LSP Traceroute to Discover Possible Loops

With the MPLS LSP Traceroute feature, loops can occur if you use the UDP destination address range option and the time-to-live option.

By default, the maximum TTL is set to 30. Therefore, the traceroute output may contain 30 lines if the target of the traceroute is not reached, which can happen when an LSP problem exists. If an LSP problem occurs, there may be duplicate entries. The router address of the last point that the trace reaches is repeated until the output is 30 lines. You can ignore the duplicate entries.

SUMMARY STEPS

1. **enable**
2. **trace mpls ipv4** *destination-address /destination-mask* [**destination** *address-start address-end address increment*] [**ttl** *maximum-time-to-live*]
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**trace mpls ipv4** *destination-address* /*destination-mask* [**destination** *address-start address-end address increment*] [**ttl** *maximum-time-to-live*] **Example:** `Router# trace mpls ipv4 10.131.159.251/32 destination 127.0.0.1 127.0.0.3 1 ttl 5`	Discovers MPLS LSP routes that packets take when traveling to their destinations. The example shows how a loop can occur.
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

What to Do Next

Tracking Packets Tagged as Implicit Null

SUMMARY STEPS

1. **enable**
2. **trace mpls ipv4** *destination-address*/*destination-mask*
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**trace mpls ipv4** *destination-address*/*destination-mask* **Example:** `Router# trace mpls ipv4 10.131.159.252/32`	Discovers MPLS LSP routes that packets actually take when traveling to their destinations.

	Command or Action	Purpose
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Tracking Untagged Packets

SUMMARY STEPS

1. **enable**
2. **show mpls forwarding-table** *destination-address/destination-mask*
3. **show mpls ldp discovery**
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show mpls forwarding-table** *destination-address/destination-mask* **Example:** `Router# show mpls forwarding-table 10.131.159.252/32`	Displays the content of the MPLS Label Forwarding Information Base (LFIB) and displays whether the LDP is properly configured.
Step 3	**show mpls ldp discovery** **Example:** `Router# show mpls ldp discovery`	Displays the status of the LDP discovery process and displays whether the LDP is properly configured.
Step 4	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Determining Why a Packet Could Not Be Sent

The Q return code means that the packet could not be sent. The problem can be caused by insufficient processing memory, but it probably results because an LSP could not be found that matches the FEC information that was entered on the command line.

You need to determine the reason why the packet was not forwarded so that you can fix the problem in the path of the LSP. To do so, look at the Routing Information Base (RIB), the Forwarding Information Base (FIB), the Label Information Base (LIB), and the MPLS LFIB. If there is no entry for the FEC in any of these routing or forwarding bases, there is a Q return code.

To determine why a packet could not be transmitted, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show ip route** [*ip-address* [**mask**]]
3. **show mpls forwarding-table** [*network* {*mask* | *length*} | **labels** *label*[-*label*] | **interface** *interface* | **next-hop** *address* | **lsp-tunnel** [*tunnel-id*]]
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip route** [*ip-address* [**mask**]] **Example:** `Router# show ip route 10.0.0.1`	Displays the current state of the routing table. When the MPLS echo reply returns a Q, troubleshooting occurs on the routing information database.
Step 3	**show mpls forwarding-table** [*network* {*mask* | *length*} | **labels** *label*[-*label*] | **interface** *interface* | **next-hop** *address* | **lsp-tunnel** [*tunnel-id*]] **Example:** `Router# show mpls forwarding-table 10.0.0.1/32`	Displays the content of the MPLS LFIB. When the MPLS echo reply returns a Q, troubleshooting occurs on a label information database and an MPLS forwarding information database.

	Command or Action	Purpose
Step 4	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Detecting LSP Breaks when Load Balancing Is Enabled for IPv4 LDP LSPs

An ICMP ping or trace follows one path from the originating router to the target router. Round robin load balancing of IP packets from a source router discovers the various output paths to the target IP address.

For MPLS ping and traceroute, Cisco routers use the source and destination addresses in the IP header for load balancing when multiple paths exist through the network to a target router. The Cisco implementation of MPLS may check the destination address of an IP payload to accomplish load balancing (the type of checking depends on the platform).

To detect LSP breaks when load balancing is enabled for IPv4 LDP LSPs, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **ping mpls ipv4** *destination-address/destination-mask-length* [**destination** *address-start address-end increment*]
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**ping mpls ipv4** *destination-address/destination-mask-length* [**destination** *address-start address-end increment*] **Example:** `Router# ping mpls ipv4 10.131.159.251/32 destination 127.0.0.1/8`	Checks for load balancing paths. Enter the 127.*z.y.x* /8 destination address.

	Command or Action	Purpose
Step 3	**exit** **Example:** `Router#` **exit**	Returns to user EXEC mode.

Specifying the Interface Through Which Echo Packets Leave a Router

You can control the interface through which packets leave a router. Path output information is used as input to LSP ping and traceroute.

The echo request output interface control feature allows you to force echo packets through the paths that perform detailed debugging or characterizing of the LSP. This feature is useful if a PE router connects to an MPLS cloud and there are broken links. You can direct traffic through a certain link. The feature also is helpful for troubleshooting network problems.

To specify the output interface for echo requests, perform the following steps.

SUMMARY STEPS

1. **enable**
2. Enter one of the following commands:
    - **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**output interface** *tx-interface*]
    - **trace mpls ipv4** *destination-address/destination-mask*
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Enter one of the following commands: • **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**output interface** *tx-interface*]	Checks MPLS LSP connectivity. or Discovers MPLS LSP routes that packets actually take when traveling to their destinations.

	Command or Action	Purpose

	• **trace mpls ipv4** *destination-address/destination-mask*
	
	Example:
	`Router# ping mpls ipv4 10.131.159.251/32 output interface fastethernet0/0/0`
	
	Example:
	`Router# trace mpls ipv4 10.131.159.251/32 output interface fastethernet0/0/0`
		Note For this task, you must specify the **output interface** keyword.

Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Pacing the Transmission of Packets

Echo request traffic pacing allows you to pace the transmission of packets so that the receiving router does not drop packets. To perform echo request traffic pacing, perform the following steps.

SUMMARY STEPS

1. **enable**
2. Do one of the following:
    - **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**interval** *ms*]]
    - **trace mpls ipv4** *destination-address/destination-mask*
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose

Step 2	Do one of the following:
	• **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**interval** *ms*]]
	• **trace mpls ipv4** *destination-address/destination-mask*
	Example:
	`Router# ping mpls ipv4 10.131.159.251/32 interval 2`
	Example:
	`Router# trace mpls ipv4 10.131.159.251/32`
		Checks MPLS LSP connectivity.
		or
		Discovers MPLS LSP routes that packets take when traveling to their destinations.
		Note In this task, if you use the **ping mpls** command you must specify the **interval** keyword.

Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Interrogating the Transit Router for Its Downstream Information by Using Echo Request request-dsmap

The echo request request-dsmap capability troubleshooting feature, used in conjunction with the TTL flag, allows you to selectively interrogate a transit router. If there is a failure, you do not have to enter an **lsp traceroute** command for each previous failure; you can focus just on the failed hop.

A request-dsmap flag in the downstream mapping flags field, and procedures that specify how to trace noncompliant routers allow you to arbitrarily time-to-live (TTL) expire MPLS echo request packets with a wildcard downstream map (DSMAP).

Echo request DSMAPs received without labels indicate that the sender did not have any DSMAPs to validate. If the downstream router ID field of the DSMAP TLV in an echo request is set to the ALLROUTERs address (224.0.0.2) and there are no labels, the source router can arbitrarily query a transit router for its DSMAP information.

The **ping mpls** command allows an MPLS echo request to be TTL-expired at a transit router with a wildcard DSMAP for the explicit purpose of troubleshooting and querying the downstream router for its DSMAPs. The default is that the DSMAP has an IPv4 bitmap hashkey. You also can select hashkey 0 (none). The purpose of the **ping mpls** command is to allow the source router to selectively TTL expire an echo request at a transit router to interrogate the transit router for its downstream information. The ability to also select a multipath (hashkey) type allows the transmitting router to interrogate a transit router for load-balancing information as is done with multipath LSP traceroute, but without having to interrogate all subsequent nodes traversed between the source router and the router on which each echo request TTL expires. Use an echo request in conjunction with the TTL setting because if an echo request arrives at the egress of the LSP with an echo request, the responding routers never return DSMAPs.

To interrogate the transit router for its downstream information so that you can focus just on the failed hop if there is a failure, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**dsmap** [**hashkey** {**none** | **ipv4 bitmap** *bitmap-size*}]]
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} [**dsmap** [**hashkey** {**none** | **ipv4 bitmap** *bitmap-size*}]] **Example:** `Router# ping mpls ipv4 10.161.251/32 dsmap hashkey ipv4 bitmap 16`	Checks MPLS LSP connectivity. **Note** In this task, you must specify the **dsmap** and **hashkey** keywords.
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Interrogating a Router for Its DSMAP

The router can interrogate the software or hardware forwarding layer for the depth limit that needs to be returned in the DSMAP TLV. If forwarding does not provide a value, the default is 255.

To determine the depth limit, specify the **dsmap** and **ttl** keywords in the **ping mpls** command. The transit router will be interrogated for its DSMAP. The depth limit is returned with the echo reply DSMAP. A value of 0 means that the IP header is used for load balancing. Another value indicates that the IP header load balances up to the specified number of labels.

To interrogate a router for its DSMAP, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **ttl** *time-to-live* **dsmap**
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **ttl** *time-to-live* **dsmap** **Example:** `Router# ping mpls ipv4 10.131.159.252/32 ttl 1 dsmap`	Checks MPLS LSP connectivity. **Note** You must specify the **ttl** and **dsmap** keywords.
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Requesting that a Transit Router Validate the Target FEC Stack

An MPLS echo request tests a particular LSP. The LSP to be tested is identified by the FEC stack.

To request that a transit router validate the target FEC stack, set the V flag from the source router by entering the **flags fec** keyword in the **ping mpls** and **trace mpls** commands. The default is that echo request packets are sent with the V flag set to 0.

To request that a transit router validate the target FEC stack, perform the following steps.

SUMMARY STEPS

1. **enable**
2. Do one of the following:
   - **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **flags fec**
   - **trace mpls ipv4** *destination-address/destination-mask* **flags fec**
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	Do one of the following:
	• **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **flags fec**
	• **trace mpls ipv4** *destination-address/destination-mask* **flags fec**
	
	Example:
	
	`Router# ping mpls ipv4 10.131.159.252/32 flags fec`
	
	Example:
	
	`Router# trace mpls ipv4 10.131.159.252/32 flags fec`
		Checks MPLS LSP connectivity.
		or
		Discovers MPLS LSP routes that packets actually take when traveling to their destinations.
		Note You must enter the **flags fec** keyword.

Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Enabling LSP Ping to Detect LSP Breakages Caused by Untagged Interfaces

For MPLS LSP ping and traceroute of LSPs carrying IPv4 FECs, you can force an explicit null label to be added to the MPLS label stack even though the label was unsolicited. This allows LSP ping to detect LSP breakages caused by untagged interfaces. LSP ping does not report that an LSP is operational when it is unable to send MPLS traffic.

An explicit null label is added to an MPLS label stack if MPLS echo request packets are forwarded from untagged interfaces that are directly connected to the destination of the LSP ping or if the IP TTL value for the MPLS echo request packets is set to 1.

When you enter an **lsp ping** command, you are testing the LSP's ability to carry IP traffic. Failure at untagged output interfaces at the penultimate hop are not detected. Explicit-null shimming allows you to test an LSP's ability to carry MPLS traffic.

To enable LSP ping to detect LSP breakages caused by untagged interfaces, specify the **force-explicit-null** keyword in the **ping mpls** or **trace mpls** commands as shown in the following steps.

SUMMARY STEPS

1. **enable**
2. Do one of the following:
    - **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **force-explicit-null**
    - **trace mpls ipv4** *destination-address/destination-mask* **force-explicit-null**
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	Do one of the following:
	• **ping mpls** {**ipv4** *destination-address/destination-mask* | **pseudowire** *ipv4-address* **vc-id** *vc-id*} **force-explicit-null**
	• **trace mpls ipv4** *destination-address/destination-mask* **force-explicit-null**
	
	Example:
	
	`Router# ping mpls ipv4 10.131.191.252/32`
	`force-explicit null`
		Checks MPLS LSP connectivity.
		or
		Discovers MPLS LSP routes that packets actually take when traveling to their destinations.
		Note You must enter the **force-explicit-null** keyword.

	Command or Action	Purpose
	Example: `Router# trace mpls ipv4 10.131.191.252/32 force-explicit-null`	
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Viewing the AToM VCCV Capabilities Advertised to and Received from the Peer

To view the AToM VCCV capabilities advertised to and received from the peer, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show mpls l2transport binding**
3. **exit**

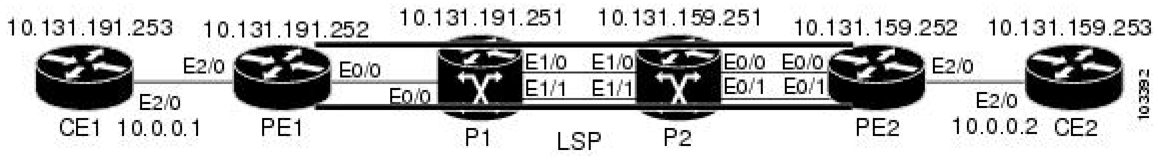
DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show mpls l2transport binding** **Example:** `Router# show mpls l2transport binding`	Displays VC label binding information.
Step 3	**exit** **Example:** `Router# exit`	Returns to user EXEC mode.

Configuration Examples for MPLS LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

Examples for the MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV feature are based on the sample topology shown in the figure below.

Figure 5: Sample Topology for Configuration Examples



Enabling Compatibility Between the MPLS LSP and Ping or Traceroute Implementation Example

The following example shows how to configure MPLS multipath LSP traceroute to interoperate with a vendor implementation that does not interpret RFC 4379 as Cisco does:

```
configure terminal
!
mpls oam
 echo revision 4
 no echo vendor-extension
 exit
```

The default echo revision number is 4, which corresponds to the IEFT draft 11.

Validating an LDP IPv4 FEC by Using MPLS LSP Ping and MPLS LSP Traceroute Example

The following example shows how to use the **ping mpls** command to test connectivity of an IPv4 LDP LSP:

```
Router# ping mpls ipv4 10.131.191.252/32 repeat 5 exp 5 verbose
Sending 5, 100-byte MPLS Echos to 10.131.191.252, timeout is 2 seconds:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!     10.131.191.230, return code 3
!     10.131.191.230, return code 3
!     10.131.191.230, return code 3
!     10.131.191.230, return code 3
!     10.131.191.230, return code 3
Success rate is 100 percent (5/5), round-trip min/avg/max = 100/10
```

Validating a Layer 2 FEC by Using MPLS LSP Ping Example

The following example validates a Layer 2 FEC:

```
Router# ping mpls pseudowire 10.10.10.15 108 vc-id 333

Sending 5, 100-byte MPLS Echos to 10.10.10.15,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
   'L' - labeled output interface, 'B' - unlabeled output interface,
   'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
   'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
   'P' - no rx intf label prot, 'p' - premature termination of LSP,
   'R' - transit router, 'I' - unknown upstream index,
   'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/32/40 ms PE-802#
```

Using DSCP to Request a Specific Class of Service in an Echo Reply Example

The following example shows how to use DSCP to request a specific CoS in an echo reply:

```
Router# ping mpls ipv4 10.131.159.252/32 reply dscp 50
 <0-63>   Differentiated services codepoint value
 af11     Match packets with AF11 dscp (001010)
 af12     Match packets with AF12 dscp (001100)
 af13     Match packets with AF13 dscp (001110)
 af21     Match packets with AF21 dscp (010010)
 af22     Match packets with AF22 dscp (010100)
 af23     Match packets with AF23 dscp (010110)
 af31     Match packets with AF31 dscp (011010)
 af32     Match packets with AF32 dscp (011100)
 af33     Match packets with AF33 dscp (011110)
 af41     Match packets with AF41 dscp (100010)
 af42     Match packets with AF42 dscp (100100)
 af43     Match packets with AF43 dscp (100110)
 cs1      Match packets with CS1(precedence 1) dscp (001000)
 cs2      Match packets with CS2(precedence 2) dscp (010000)
 cs3      Match packets with CS3(precedence 3) dscp (011000)
 cs4      Match packets with CS4(precedence 4) dscp (100000)
 cs5      Match packets with CS5(precedence 5) dscp (101000)
 cs6      Match packets with CS6(precedence 6) dscp (110000)
 cs7      Match packets with CS7(precedence 7) dscp (111000)
 default  Match packets with default dscp (000000)
 ef       Match packets with EF dscp (101110)
```

Controlling How a Responding Router Replies to an MPLS Echo Request Example

The following example checks MPLS LSP connectivity by using ipv4 reply mode:

```
Router# ping mpls ipv4 10.131.191.252/32 reply mode ipv4
```

Preventing Possible Loops with MPLS LSP Ping Example

The following example shows how a loop operates if you use the following **ping mpls** command:

```
Router# ping mpls
 ipv4
 10.131.159.251/32 destination 127.0.0.1 127.0.0.2 1 repeat 2
sweep 1450 1475 25
Sending 2, [1450..1500]-byte MPLS Echos to 10.131.159.251/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
Destination address 127.0.0.1
!
!
Destination address 127.0.0.2
!
!
Destination address 127.0.0.1
!
!
Destination address 127.0.0.2
!
!
```

A **ping mpls** command is sent for each packet size range for each destination address until the end address is reached. For this example, the loop continues in the same manner until the destination address, 127.0.0.5, is reached. The sequence continues until the number is reached that you specified with the **repeat** *count* keyword and argument. For this example, the repeat count is 2. The MPLS LSP ping loop sequence is as follows:

```
repeat  = 1
  destination address 1 (address-start
)
    for (size from sweep minimum
 to maximum
, counting by size-increment
)
      send an lsp ping
  destination address 2 (address-start
 +
address-
increment
)
    for (size from sweep minimum
 to maximum
, counting by size-increment
)
      send an lsp ping
  destination address 3 (address-start
 +
address-
increment
 +
address-
increment
)
     for (size from sweep minimum
 to maximum
, counting by size-increment
)
```

```
        send an lsp ping
    .
    .
    .
    until destination address = address-end
    .
    .
    .
    until repeat = count 2
```

Preventing Possible Loops with MPLS LSP Traceroute Example

The following example shows how a loop occurs if you use the following **trace mpls** command:

```
Router# trace mpls ipv4 10.131.159.251/32 destination 127.0.0.1 127.0.0.3 1 ttl 5
Tracing MPLS Label Switched Path to 10.131.159.251/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
Destination address 127.0.0.1
  0 10.131.191.230 MRU 1500 [Labels: 19 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
! 2 10.131.159.225 40 ms
Destination address 127.0.0.2
  0 10.131.191.230 MRU 1500 [Labels: 19 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
! 2 10.131.159.225 40 ms
Destination address 127.0.0.3
  0 10.131.191.230 MRU 1500 [Labels: 19 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
! 2 10.131.159.225 48 ms
```

An **mpls trace** command is sent for each TTL from 1 to the maximum TTL (**ttl** *maximum-time-to-live* keyword and argument) for each destination address until the address specified with the destination *end-address* argument is reached. In this example, the maximum TTL is 5 and the end destination address is 127.0.0.3. The MPLS LSP traceroute loop sequence is as follows:

```
destination address 1 (address-start
)
  for (ttl from 1 to maximum-time-to-live
)
    send an lsp trace
destination address 2 (address-start
 + address-increment
)
  for (ttl from 1 to 5
)
    send an lsp trace
destination address 3 (address-start
 + address-increment
 + address-increment
)
  for (ttl from 1 to
maximum-time-to-live)
    send an lsp trace
.
.
.
until destination address = 4
```

The following example shows that the trace encountered an LSP problem at the router that has an IP address of 10.6.1.6:

```
Router# traceroute mpls ipv4 10.6.7.4/32
Tracing MPLS Label Switched Path to 10.6.7.4/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.6.1.14 MRU 4470 [Labels: 22 Exp: 0]
R 1 10.6.1.5 MRU 4470 [Labels: 21 Exp: 0] 2 ms
R 2 10.6.1.6 4 ms                  <------ Router address repeated for 2nd to 30th TTL.
R 3 10.6.1.6 1 ms
R 4 10.6.1.6 1 ms
R 5 10.6.1.6 3 ms
R 6 10.6.1.6 4 ms
R 7 10.6.1.6 1 ms
R 8 10.6.1.6 2 ms
R 9 10.6.1.6 3 ms
R 10 10.6.1.6 4 ms
R 11 10.6.1.6 1 ms
R 12 10.6.1.6 2 ms
R 13 10.6.1.6 4 ms
R 14 10.6.1.6 5 ms
R 15 10.6.1.6 2 ms
R 16 10.6.1.6 3 ms
R 17 10.6.1.6 4 ms
R 18 10.6.1.6 2 ms
R 19 10.6.1.6 3 ms
R 20 10.6.1.6 4 ms
R 21 10.6.1.6 1 ms
R 22 10.6.1.6 2 ms
R 23 10.6.1.6 3 ms
R 24 10.6.1.6 4 ms
R 25 10.6.1.6 1 ms
R 26 10.6.1.6 3 ms
R 27 10.6.1.6 4 ms
R 28 10.6.1.6 1 ms
R 29 10.6.1.6 2 ms
R 30 10.6.1.6 3 ms                  <------ TTL 30.
```

If you know the maximum number of hops in your network, you can set the TTL to a lower value with the **trace mpls ttl** *maximum-time-to-live* command. The following example shows the same **traceroute** command as the previous example, except that this time the TTL is set to 5:

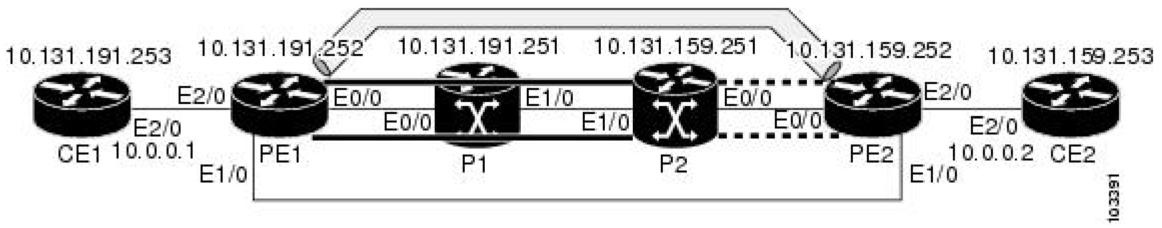
```
Router# traceroute mpls ipv4 10.6.7.4/32 ttl 5
Tracing MPLS Label Switched Path to 10.6.7.4/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.6.1.14 MRU 4470 [Labels: 22 Exp: 0]
R 1 10.6.1.5 MRU 4474 [No Label] 3 ms
R 2 10.6.1.6 4 ms                  <------ Router address repeated for 2nd to 5th TTL.
R 3 10.6.1.6 1 ms
R 4 10.6.1.6 3 ms
R 5 10.6.1.6 4 ms
```

Troubleshooting with LSP Ping or Traceroute Example

ICMP **ping** and **trace** commands are often used to help diagnose the root cause of a failure. When an LSP is broken, the packet may reach the target router by IP forwarding, thus making the ICMP ping and traceroute features unreliable for detecting MPLS forwarding problems. The MPLS LSP ping or traceroute and AToM VCCV features extend this diagnostic and troubleshooting ability to the MPLS network and handle inconsistencies (if any) between the IP and MPLS forwarding tables, inconsistencies in the MPLS control and data plane, and problems with the reply path.

The figure below shows a sample topology with an LDP LSP.

Figure 6: Sample Topology with LDP LSP



This section contains the following subsections:

Configuration for Sample Topology

These are sample topology configurations for the troubleshooting examples in the following sections (see the figure above). There are the six sample router configurations.

Router CE1 Configuration

Following is the configuration for the CE1 router:

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname CE1
!
boot-start-marker
boot-end-marker
!
enable password lab
!
clock timezone EST -5
ip subnet-zero
!
!
!
interface Loopback0
 ip address 10.131.191.253 255.255.255.255
 no ip directed-broadcast
 no clns route-cache
!
!
interface Ethernet2/0
 no ip address
 no ip directed-broadcast
 no keepalive
 no cdp enable
 no clns route-cache
!
interface Ethernet2/0.1
 encapsulation dot1Q 1000
 ip address 10.0.0.1 255.255.255.0
 no ip directed-broadcast
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
end
```

Router PE1 Configuration

Following is the configuration for the PE1 router:

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname PE1
!
boot-start-marker
boot-end-marker
!
logging snmp-authfail
enable password lab
!
clock timezone EST -5
ip subnet-zero
ip cef
no ip domain-lookup
!
mpls ldp discovery targeted-hello accept
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
!
!
interface Loopback0
 ip address 10.131.191.252 255.255.255.255
 no clns route-cache
!
interface Ethernet0/0
 ip address 10.131.191.230 255.255.255.252
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
interface Ethernet1/0
 ip address 10.131.159.246 255.255.255.252
 shutdown
 no clns route-cache
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
interface Ethernet2/0
 no ip address
 no cdp enable
 no clns route-cache
!
```

```
interface Ethernet2/0.1
 encapsulation dot1Q 1000
 xconnect 10.131.159.252 333 encapsulation mpls
!
!
router ospf 1
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.159.244 0.0.0.3 area 0
 network 10.131.191.228 0.0.0.3 area 0
 network 10.131.191.232 0.0.0.3 area 0
 network 10.131.191.252 0.0.0.0 area 0
!
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
!
end
```

Router P1 Configuration

Following is the configuration for the P1 router:

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname P1
!
boot-start-marker
boot-end-marker
!
logging snmp-authfail
enable password lab
!
clock timezone EST -5
ip subnet-zero
ip cef
no ip domain-lookup
!
!
mpls ldp discovery targeted-hello accept
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
!
!
 no clns route-cache
!
interface Loopback0
 ip address 10.131.191.251 255.255.255.255
 no clns route-cache
!
interface Ethernet0/0
 ip address 10.131.191.229 255.255.255.252
 no clns route-cache
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
interface Ethernet1/0
 ip address 10.131.159.226 255.255.255.252
 no clns route-cache
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
interface Ethernet1/1
 ip address 10.131.159.222 255.255.255.252
 no clns route-cache
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
!
router ospf 1
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.159.220 0.0.0.3 area 0
 network 10.131.159.224 0.0.0.3 area 0
 network 10.131.191.228 0.0.0.3 area 0
 network 10.131.191.251 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
end
```

Router P2 Configuration

Following is the configuration for the P2 router:

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname P2
!
boot-start-marker
boot-end-marker
!
enable password lab
!
clock timezone EST -5
ip subnet-zero
ip cef
no ip domain-lookup
!
mpls ldp discovery targeted-hello accept
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
!
!
interface Loopback0
 ip address 10.131.159.251 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 10.131.159.229 255.255.255.252
 no ip directed-broadcast
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
interface Ethernet0/1
 ip address 10.131.159.233 255.255.255.252
no ip directed-broadcast
```

```
ip rsvp signalling dscp 0
!
interface Ethernet1/0
 ip address 10.131.159.225 255.255.255.252
no ip directed-broadcast
ip rsvp bandwidth 1500 1500
ip rsvp signalling dscp 0
!
interface Ethernet1/1
 ip address 10.131.159.221 255.255.255.252
no ip directed-broadcast
ip rsvp signalling dscp 0
!
!
router ospf 1
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.159.220 0.0.0.3 area 0
 network 10.131.159.224 0.0.0.3 area 0
 network 10.131.159.228 0.0.0.3 area 0
 network 10.131.159.232 0.0.0.3 area 0
 network 10.131.159.251 0.0.0.0 area 0
!
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
end
```

Router PE2 Configuration

Following is the configuration for the PE2 router:

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname PE2
!
boot-start-marker
boot-end-marker
!
logging snmp-authfail
enable password lab
!
clock timezone EST -5
ip subnet-zero
ip cef
no ip domain-lookup
!
mpls ldp discovery targeted-hello accept
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
!
!
interface Loopback0
 ip address 10.131.159.252 255.255.255.255
 no clns route-cache
!
interface Ethernet0/0
 ip address 10.131.159.230 255.255.255.252
 no clns route-cache
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface Ethernet0/1
 ip address 10.131.159.234 255.255.255.252
 no clns route-cache
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface Ethernet1/0
 ip address 10.131.159.245 255.255.255.252
 mpls ip
 no clns route-cache
!
interface Ethernet3/0
 no ip address
 no cdp enable
 no clns route-cache
!
interface Ethernet3/0.1
 encapsulation dot1Q 1000
 no snmp trap link-status
 no cdp enable
 xconnect 10.131.191.252 333 encapsulation mpls
!
!
router ospf 1
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.122.0 0.0.0.3 area 0
 network 10.131.159.228 0.0.0.3 area 0
 network 10.131.159.232 0.0.0.3 area 0
 network 10.131.159.236 0.0.0.3 area 0
 network 10.131.159.244 0.0.0.3 area 0
 network 10.131.159.252 0.0.0.0 area 0
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
!
end
```

Router CE2 Configuration

Following is the configuration for the CE2 router:

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname CE2
!
boot-start-marker
boot-end-marker
!
enable password lab
!
clock timezone EST -5
ip subnet-zero
ip cef
no ip domain-lookup
!
!
```

```
interface Loopback0
 ip address 10.131.159.253 255.255.255.255
 no ip directed-broadcast
 no clns route-cache
!
interface Ethernet3/0
 no ip address
 no ip directed-broadcast
 no keepalive
 no cdp enable
 no clns route-cache
!
interface Ethernet3/0.1
 encapsulation dot1Q 1000
 ip address 10.0.0.2 255.255.255.0
 no ip directed-broadcast
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
end
```

Verification That the LSP Is Configured Correctly

Use the output from the **show** commands in this section to verify that the LSP is configured correctly.

A **show mpls forwarding-table** command shows that tunnel 1 is in the MPLS forwarding table.

```
PE1# show mpls forwarding-table 10.131.159.252
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
22     18
[T] 10.131.159.252/32 0          Tu1        point2point
[T]     Forwarding through a TSP tunnel.
        View additional tagging info with the 'detail' option
```

A **trace mpls** command issued at PE1 verifies that packets with 16 as the outermost label and 18 as the end-of-stack label are forwarded from PE1 to PE2.

```
PE1# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
   0 10.131.191.252 MRU 1496 [Labels: 16/18 Exp: 0/0] L 1 10.131.191.229
   MRU 1508 [Labels: 18 Exp: 0] 0 ms L 2 10.131.159.225
   MRU 1504 [Labels: implicit-null Exp: 0] 0 ms ! 3 10.131.159.234 20 ms
   PE1#
```

The MPLS LSP Traceroute to PE2 is successful, as indicated by the exclamation point (!).

Discovery of LSP Breaks

Use the output of the commands in this section to discover LSP breaks.

An LDP target session is established between routers PE1 and P2, as shown in the output of the following **show mpls ldp discovery** command:

```
PE1# show mpls ldp discovery
 Local LDP Identifier:
    10.131.191.252:0
    Discovery Sources:
    Interfaces:
        Ethernet0/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
        Tunnel1 (ldp): Targeted -> 10.131.159.251
    Targeted Hellos:
        10.131.191.252 -> 10.131.159.252 (ldp): active/passive, xmit/recv
            LDP Id: 10.131.159.252:0
        10.131.191.252 -> 10.131.159.251 (ldp): active, xmit/recv
LDP Id: 10.131.159.251:0
```

Enter the following command on the P2 router in global configuration mode:

```
P2(config)# no mpls ldp discovery targeted-hello accept
```

The LDP configuration change causes the targeted LDP session between the headend and tailend of the TE tunnel to go down. Labels for IPv4 prefixes learned by P2 are not advertised to PE1. Thus, all IP prefixes reachable by P2 are reachable by PE1 only through IP (not MPLS). In other words, packets destined for those prefixes through Tunnel 1 at PE1 will be IP switched at P2 (which is undesirable).

The following **show mpls ldp discovery** command shows that the LDP targeted session is down:

```
PE1# show mpls ldp discovery
 Local LDP Identifier:
    10.131.191.252:0
    Discovery Sources:
    Interfaces:
        Ethernet0/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
        Tunnel1 (ldp): Targeted -> 10.131.159.251
    Targeted Hellos:
        10.131.191.252 -> 10.131.159.252 (ldp): active/passive, xmit/recv
            LDP Id: 10.131.159.252:0
        10.131.191.252 -> 10.131.159.251 (ldp): active, xmit
```

Enter the **show mpls forwarding-table** command at the PE1 router. The display shows that the outgoing packets are untagged as a result of the LDP configuration changes.

```
PE1# show mpls forwarding-table 10.131.159.252
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
22     Untagged[T] 10.131.159.252/32 0          Tu1        point2point
[T]     Forwarding through a TSP tunnel.
        View additional tagging info with the 'detail' option
```

A **ping mpls** command entered at the PE1 router displays the following:

```
PE1# ping mpls ipv4 10.131.159.252/32 repeat 1
Sending 1, 100-byte MPLS Echos to 10.131.159.252/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
R
Success rate is 0 percent (0/1)
```

The **ping mpls** command fails. The R indicates that the sender of the MPLS echo reply had a routing entry but no MPLS FEC. Entering the **verbose** keyword with the **ping mpls** command displays the MPLS LSP echo reply sender address and the return code. You should be able to determine where the breakage occurred by telnetting to the replying router and inspecting its forwarding and label tables. You might need to look at the neighboring upstream router as well, because the breakage might be on the upstream router.

```
PE1# ping mpls ipv4 10.131.159.252/32 repeat 1 verbose
Sending 1, 100-byte MPLS Echos to 10.131.159.252/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
R   10.131.159.225, return code 6
Success rate is 0 percent (0/1)
```

Alternatively, use the LSP **traceroute** command to figure out which router caused the breakage. In the following example, for subsequent values of TTL greater than 2, the same router keeps responding (10.131.159.225). This suggests that the MPLS echo request keeps getting processed by the router regardless of the TTL. Inspection of the label stack shows that P1 pops the last label and forwards the packet to P2 as an IP packet. This explains why the packet keeps getting processed by P2. MPLS echo request packets cannot be forwarded by use of the destination address in the IP header because the address is set to a 127/8 address.

```
PE1# trace mpls ipv4 10.131.159.252/32 ttl 5
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.191.230 MRU 1496 [Labels: 22/19 Exp: 0/0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 40 ms
R 2 10.131.159.229 MRU 1504 [implicit-null] 28 ms
! 3 10.131.159.230 40 ms
pe1#
```

MTU Discovery in an LSP Example

The following example shows the results of a **trace mpls** command when the LSP is formed with labels created by LDP:

```
PE1# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.191.230 MRU 1496 [Labels: 22/19 Exp: 0/0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 40 ms
R 2 10.131.159.229 MRU 1504 [implicit-null] 28 ms
! 3 10.131.159.230 40 ms
pe1#
```

You can determine the MRU for the LSP at each hop through the use of the **show mpls forwarding detail**command:

```
PE1# show mpls forwarding 10.131.159.252 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
22     19          10.131.159.252/32 0          Tu1        point2point
        MAC/Encaps=14/22, MRU=1496, Tag Stack{22 19}, via Et0/0
        AABBCC009700AABBCC0098008847 0001600000013000
        No output feature configured
```

To determine how large an echo request will fit on the LSP, first calculate the size of the IP MTU by using the **show interface** *interface-name* command:

```
PE1# show interface e0/0
Ethernet0/0 is up, line protocol is up
  Hardware is Lance, address is aabb.cc00.9800 (bia aabb.cc00.9800)
  Internet address is 10.131.191.230/30
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:01, output 00:00:01, output hang never
  Last clearing of “show interface” counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     377795 packets input, 33969220 bytes, 0 no buffer
     Received 231137 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     441772 packets output, 40401350 bytes, 0 underruns
     0 output errors, 0 collisions, 10 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

The IP MTU in the **show interface** *interface-name* example is 1500 bytes. Subtract the number of bytes corresponding to the label stack from the MTU number. The output of the **show mpls forwarding** command indicates that the Tag stack consists of one label (21). Therefore, the largest MPLS echo request packet that can be sent in the LSP is 1500 - (2 x 4) = 1492.

You can validate this by using the following **mpls ping** command:

```
PE1# ping mpls ipv4 10.131.159.252/32 sweep 1492 1500 1 repeat 1
Sending 1, [1492..1500]-byte MPLS Echos to 10.131.159.252/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!QQQQQQQQ
Success rate is 11 percent (1/9), round-trip min/avg/max = 40/40/40 ms
```

In this command, echo packets that have a range in size from 1492 to 1500 bytes are sent to the destination address. Only packets of 1492 bytes are sent successfully, as indicated by the exclamation point (!). Packets of byte sizes 1493 to 1500 are source-quenched, as indicated by the Qs.

You can pad an MPLS echo request so that a payload of a given size can be tested. The pad TLV is useful when you use the MPLS echo request to discover the MTU that is supportable by an LSP. MTU discovery is extremely important for applications like AToM that contain non-IP payloads that cannot be fragmented.

Tracking Packets Tagged as Implicit Null Example

In the following example, Tunnel 1 is shut down, and only an LSP formed with LDP labels is established. An implicit null is advertised between the P2 and PE2 routers. Entering an MPLS LSP traceroute command at the PE1 router results in the following output that shows that packets are forwarded from P2 to PE2 with an implicit-null label. Address 10.131.159.229 is configured for the P2 Ethernet 0/0 out interface for the PE2 router.

```
PE1# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.191.230 MRU 1496 [Labels: 22/19 Exp: 0/0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 40 ms
R 2 10.131.159.229 MRU 1504 [implicit-null] 28 ms
! 3 10.131.159.230 40 ms
pe1#
```

Tracking Untagged Packets Example

Untagged cases are valid configurations for IGP LSPs that could cause problems for MPLS VPNs.

A **show mpls forwarding-table** command and a **show mpls ldp discovery** command issued at the P2 router show that LDP is properly configured:

```
P2# show mpls forwarding-table 10.131.159.252
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
19     Pop tag     10.131.159.252/32 0          Et0/0      10.131.159.230
P2# show mpls ldp discovery
 Local LDP Identifier:
    10.131.159.251:0
    Discovery Sources:
    Interfaces:
        Ethernet0/0 (ldp): xmit/recv
            LDP Id: 10.131.159.252:0
        Ethernet1/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
```

The **show mpls ldp discovery** command output shows that Ethernet interface 0/0, which connects PE2 to P2, is sending and receiving packets.

If a **no mpls ip** command is entered on Ethernet interface 0/0, this could prevent an LDP session between the P2 and PE2 routers from being established. A **show mpls ldp discovery** command entered on the PE router shows that the MPLS LDP session with the PE2 router is down.

```
P2# show mpls ldp discovery
 Local LDP Identifier:
    10.131.159.251:0
    Discovery Sources:
    Interfaces:
        Ethernet0/0 (ldp): xmit
        Ethernet1/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
```

If the MPLS LDP session to PE2 goes down, the LSP to 10.131.159.252 becomes untagged, as shown by the **show mpls forwarding-table** command:

```
P2# show mpls forwarding-table 10.131.159.252/32
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
19     Untagged    10.131.159.252/32 864        Et0/0      10.131.159.230
```

Untagged cases would provide an MPLS LSP traceroute reply with packets tagged with No Label, as shown in the following display. You may need to reestablish an MPLS LSP session from interface P2 to PE2 by entering an **mpls ip** command on the output interface from P2 to PE2, which is Ethernet 0/0 in this example:

```
PE1# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.191.230 MRU 1500 [Labels: 20 Exp: 0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 80 ms
R 2 10.131.159.229 MRU 1504 [No Label] 28 ms       <----No MPLS session from P2 to PE2.
! 3 10.131.159.230 40 ms
```

Determining Why a Packet Could Not Be Sent Example

The following example shows a **ping mpls** command when an MPLS echo request is not sent. The transmission failure is shown by the returned Qs.

```
PE1# ping mpls ipv4 10.0.0.1/32
Sending 5, 100-byte MPLS Echos to 10.0.0.1/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
QQQQQ
Success rate is 0 percent (0/5)
```

The following **show mpls forwarding-table** command and **show ip route** command demonstrate that the IPv4 address (10.0.0.1)address is not in the LFIB or RIB routing table. Therefore, the MPLS echo request is not sent.

```
PE1# show mpls forwarding-table 10.0.0.1
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
PE1# show ip route 10.0.0.1
% Subnet not in table
```

Detecting LSP Breaks when Load Balancing Is Enabled for IPv4 LSPs Example

In the following examples, different paths are followed to the same destination. The output from these examples demonstrates that load balancing occurs between the originating router and the target router.

To ensure that Ethernet interface 1/0 on the PE1 router is operational, enter the following commands on the PE1 router:

```
PE1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
PE1(config)# interface ethernet 1/0
PE1(config-if)# no shutdown
PE1(config-if)# end
*Dec 31 19:14:10.034: %LINK-3-UPDOWN: Interface Ethernet1/0, changed state to up
*Dec 31 19:14:11.054: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet1/0,
changed state to upend
PE1#
*Dec 31 19:14:12.574: %SYS-5-CONFIG_I: Configured from console by console
*Dec 31 19:14:19.334: %OSPF-5-ADJCHG: Process 1, Nbr 10.131.159.252 on Ethernet1/0
from LOADING to FULL, Loading Done
PE1#
```

The following **show mpls forwarding-table** command displays the possible outgoing interfaces and next hops for the prefix 10.131.159.251/32:

```
PE1# show mpls forwarding-table 10.131.159.251/32
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
21     19          10.131.159.251/32 0          Et0/0      10.131.191.229
       20          10.131.159.251/32 0          Et1/0      10.131.159.245
```

The following **ping mpls** command to 10.131.159.251/32 with a destination UDP address of 127.0.0.1 shows that the selected path has a path index of 0:

```
Router# ping mpls ipv4
 10.131.159.251/32 destination
 127.0.0.1/32
Sending 1, 100-byte MPLS Echos to 10.131.159.251/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 40/40/40 ms
PE1#
*Dec 29 20:42:40.638: LSPV: Echo Request sent on IPV4 LSP, load_index 2,
pathindex 0, size 100
*Dec 29 20:42:40.638: 46 00 00 64 00 00 40 00 FF 11 9D 03 0A 83 BF FC
*Dec 29 20:42:40.638: 7F 00 00 01 94 04 00 00 0D AF 0D AF 00 4C 14 70
*Dec 29 20:42:40.638: 00 01 00 00 01 02 00 00 1A 00 00 1C 00 00 00 01
*Dec 29 20:42:40.638: C3 9B 10 40 A3 6C 08 D4 00 00 00 00 00 00 00 00
*Dec 29 20:42:40.638: 00 01 00 09 00 01 00 05 0A 83 9F FB 20 00 03 00
*Dec 29 20:42:40.638: 13 01 AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Dec 29 20:42:40.638: AB CD AB CD
*Dec 29 20:42:40.678: LSPV: Echo packet received: src 10.131.159.225,
dst 10.131.191.252, size 74
*Dec 29 20:42:40.678: AA BB CC 00 98 01 AA BB CC 00 FC 01 08 00 45 C0
*Dec 29 20:42:40.678: 00 3C 32 D6 00 00 FD 11 15 37 0A 83 9F E1 0A 83
*Dec 29 20:42:40.678: BF FC 0D AF 0D AF 00 28 D1 85 00 01 00 00 02 02
*Dec 29 20:42:40.678: 03 00 1A 00 00 1C 00 00 00 01 C3 9B 10 40 A3 6C
*Dec 29 20:42:40.678: 08 D4 C3 9B 10 40 66 F5 C3 C8
```

The following **ping mpls** command to 10.131.159.251/32 with a destination UDP address of 127.0.0.3 shows that the selected path has a path index of 1:

```
PE1# ping mpls ipv4 10.131.159.251/32 destination 127.0.0.3/32
Sending 1, 100-byte MPLS Echos to 10.131.159.251/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes:
    '!' - success, 'Q' - request not sent, '.' - timeout,
    'L' - labeled output interface, 'B' - unlabeled output interface,
    'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
    'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
    'P' - no rx intf label prot, 'p' - premature termination of LSP,
    'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 40/40/40 ms
PE1#
*Dec 29 20:43:09.518: LSPV: Echo Request sent on IPV4 LSP, load_index 13,
pathindex 1, size 100
*Dec 29 20:43:09.518: 46 00 00 64 00 00 40 00 FF 11 9D 01 0A 83 BF FC
*Dec 29 20:43:09.518: 7F 00 00 03 94 04 00 00 0D AF 0D AF 00 4C 88 58
*Dec 29 20:43:09.518: 00 01 00 00 01 02 00 00 38 00 00 1D 00 00 00 01
*Dec 29 20:43:09.518: C3 9B 10 5D 84 B3 95 84 00 00 00 00 00 00 00 00
*Dec 29 20:43:09.518: 00 01 00 09 00 01 00 05 0A 83 9F FB 20 00 03 00
*Dec 29 20:43:09.518: 13 01 AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Dec 29 20:43:09.518: AB CD AB CD
*Dec 29 20:43:09.558: LSPV: Echo packet received: src 10.131.159.229,
dst 10.131.191.252, size 74
*Dec 29 20:43:09.558: AA BB CC 00 98 01 AA BB CC 00 FC 01 08 00 45 C0
*Dec 29 20:43:09.558: 00 3C 32 E9 00 00 FD 11 15 20 0A 83 9F E5 0A 83
*Dec 29 20:43:09.558: BF FC 0D AF 0D AF 00 28 D7 57 00 01 00 00 02 02
*Dec 29 20:43:09.558: 03 00 38 00 00 1D 00 00 00 01 C3 9B 10 5D 84 B3
*Dec 29 20:43:09.558: 95 84 C3 9B 10 5D 48 3D 50 78
```

To see the actual path chosen, enter the **debug mpls lspv** command with the **packet** and **data** keywords.

Note The load balancing algorithm attempts to uniformly distribute packets across the available output paths by hashing based on the IP header source and destination addresses. The selection of the *address-start*,*address-end*, and *address-increment* arguments for the **destination** keyword may not provide the expected results.

Specifying the Interface Through Which Echo Packets Leave a Router Example

The following example tests load balancing from the upstream router:

```
Router# ping mpls ipv4 10.131.161.251/32 ttl 1 repeat 1 dsmap hashkey ipv4 bitmap 8

Sending 1, 100-byte MPLS Echos to 10.131.161.251/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
L
Echo Reply received from 10.131.131.2
  DSMAP 0, DS Router Addr 10.131.141.130, DS Intf Addr 10.131.141.130
    Depth Limit 0, MRU 1500 [Labels: 54 Exp: 0]
    Multipath Addresses:
      127.0.0.3       127.0.0.5       127.0.0.7       127.0.0.8

  DSMAP 1, DS Router Addr 10.131.141.2, DS Intf Addr 10.131.141.2
    Depth Limit 0, MRU 1500 [Labels: 40 Exp: 0]
    Multipath Addresses:
      127.0.0.1       127.0.0.2       127.0.0.4       127.0.0.6
```

The following example validates that the transit router reported the proper results by determining the Echo Reply sender address two hops away and checking the rx label advertised upstream:

```
Success rate is 0 percent (0/1)
```

```
Router# trace mpls ipv4 10.131.161.251/32 destination 127.0.0.6 ttl 2
Tracing MPLS Label Switched Path to 10.131.161.251/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.131.1 10.131.131.2 MRU 1500 [Labels: 37 Exp: 0]
L 1 10.131.131.2 10.131.141.2 MRU 1500 [Labels: 40 Exp: 0] 0 ms, ret code 8
L 2 10.131.141.2 10.131.150.2 MRU 1504 [Labels: implicit-null Exp: 0] 0 ms, ret code 8
Router#
Router# telnet 10.131.141.2
Trying 10.131.141.2 ... Open
User Access Verification
Password:
Router> en
The following example shows how the output interface
 keyword forces an LSP traceroute out Ethernet interface 0/0:
Router# show mpls forwarding-table 10.131.159.251
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
20     19            10.131.159.251/32 0             Et1/0      10.131.159.245
       18            10.131.159.251/32 0             Et0/0      10.131.191.229
Router# trace mpls ipv4 10.131.159.251/32

Tracing MPLS Label Switched Path to 10.131.159.251/32, timeout is 2 seconds
Type escape sequence to abort.
  0 10.131.159.246 MRU 1500 [Labels: 19 Exp: 0]
L 1 10.131.159.245 MRU 1504 [Labels: implicit-null Exp: 0] 4 ms
! 2 10.131.159.229 20 ms
Router# trace mpls ipv4 10.131.159.251/32 output-interface ethernet0/0
Tracing MPLS Label Switched Path to 10.131.159.251/32, timeout is 2 seconds
Type escape sequence to abort.
  0 10.131.191.230 MRU 1500 [Labels: 18 Exp: 0]
L 1 10.131.191.229 MRU 1504 [Labels: implicit-null Exp: 0] 0 ms
! 2 10.131.159.225 1 ms
```

Pacing the Transmission of Packets Example

The following example shows the pace of the transmission of packets:

```
Router# ping mpls ipv4 10.5.5.5/32 interval 100

Sending 5, 100-byte MPLS Echos to 10.5.5.5/32,
     timeout is 2 seconds, send interval is 100 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/36 ms PE-802
```

Interrogating the Transit Router for Its Downstream Information Example

The following example shows sample output when a router with two output paths is interrogated:

```
Router# ping mpls ipv4 10.161.251/32 ttl 4 repeat 1 dsmap hashkey ipv4 bitmap 16

Sending 1, 100-byte MPLS Echos to 10.131.161.251/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
L
Echo Reply received from 10.131.131.2
  DSMAP 0, DS Router Addr 10.131.141.130, DS Intf Addr 10.131.141.130
    Depth Limit 0, MRU 1500 [Labels: 54 Exp: 0]
    Multipath Addresses:
      127.0.0.3       127.0.0.6       127.0.0.9       127.0.0.10
      127.0.0.12      127.0.0.13      127.0.0.14      127.0.0.15
      127.0.0.16
  DSMAP 1, DS Router Addr 10.131.141.2, DS Intf Addr 10.131.141.2
    Depth Limit 0, MRU 1500 [Labels: 40 Exp: 0]
    Multipath Addresses:
      127.0.0.1       127.0.0.2       127.0.0.4       127.0.0.5
      127.0.0.7       127.0.0.8       127.0.0.11
Success rate is 0 percent (0/1)
```

The multipath addresses cause a packet to transit to the router with the output label stack. The **ping mpls** command is useful for determining the number of output paths, but when the router is more than one hop away a router cannot always use those addresses to get the packet to transit through the router being interrogated. This situation exists because the change in the IP header destination address may cause the packet to be load-balanced differently by routers between the source router and the responding router. Load balancing is affected by the source address in the IP header. The following example tests load-balancing reporting from the upstream router:

```
Router# ping mpls ipv4 10.131.161.251/32 ttl 1 repeat 1 dsmap hashkey ipv4 bitmap 8

Sending 1, 100-byte MPLS Echos to 10.131.161.251/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
L
Echo Reply received from 10.131.131.2
  DSMAP 0, DS Router Addr 10.131.141.130, DS Intf Addr 10.131.141.130
    Depth Limit 0, MRU 1500 [Labels: 54 Exp: 0]
    Multipath Addresses:
      127.0.0.3       127.0.0.5       127.0.0.7       127.0.0.8

  DSMAP 1, DS Router Addr 10.131.141.2, DS Intf Addr 10.131.141.2
    Depth Limit 0, MRU 1500 [Labels: 40 Exp: 0]
    Multipath Addresses:
      127.0.0.1       127.0.0.2       127.0.0.4       127.0.0.6
```

To validate that the transit router reported the proper results, determine the Echo Reply sender address that is two hops away and consistently check the rx label that is advertised upstream. The following is sample output:

```
Success rate is 0 percent (0/1)
Router# trace mpls ipv4 10.131.161.251/32 destination 127.0.0.6 ttl 2
Tracing MPLS Label Switched Path to 10.131.161.251/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.131.1 10.131.131.2 MRU 1500 [Labels: 37 Exp: 0]
```

```
L 1 10.131.131.2 10.131.141.2 MRU 1500 [Labels: 40 Exp: 0] 0 ms, ret code 8
L 2 10.131.141.2 10.131.150.2 MRU 1504 [Labels: implicit-null Exp: 0] 0 ms, ret code 8
Router#
Router# telnet 10.131.141.2

Trying 10.131.141.2 ... Open
User Access Verification
Password:
Router> en
Router# show mpls forwarding-table 10.131.161.251

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
40     Pop tag     10.131.161.251/32 268        Et1/0      10.131.150.2
Router#
```

Interrogating a Router for Its DSMAP Example

The following example interrogates the software and hardware forwarding layer for their depth limit that needs to be returned in the DSMAP TLV.

```
Router# ping mpls ipv4 10.131.159.252/32 ttl 1 dsmap
Sending 1, 100-byte MPLS Echos to 10.131.159.252/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes:
  '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
L
Echo Reply received from 10.131.191.229
  DSMAP 0, DS Router Addr 10.131.159.225, DS Intf Addr 10.131.159.225
    Depth Limit 0, MRU 1508 [Labels: 18 Exp: 0]
    Multipath Addresses:
      127.0.0.1       127.0.0.2       127.0.0.3       127.0.0.4
      127.0.0.5       127.0.0.6       127.0.0.7       127.0.0.8
      127.0.0.9       127.0.0.10     127.0.0.11     127.0.0.12
      127.0.0.13     127.0.0.14     127.0.0.15     127.0.0.16
      127.0.0.17     127.0.0.18     127.0.0.19     127.0.0.20
      127.0.0.21     127.0.0.22     127.0.0.23     127.0.0.24
      127.0.0.25     127.0.0.26     127.0.0.27     127.0.0.28
      127.0.0.29     127.0.0.30     127.0.0.31     127.0.0.32
Success rate is 0 percent (0/1)
```

Requesting that a Transit Router Validate the Target FEC Stack Example

The following example causes a transit router to validate the target FEC stack by which an LSP to be tested is identified:

```
Router# trace mpls ipv4 10.5.5.5/32 flags fec

Tracing MPLS Label Switched Path to 10.5.5.5/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.2.3.2 10.2.3.3 MRU 1500 [Labels: 19 Exp: 0] L 1 10.2.3.3 10.3.4.4 MRU 1500 [Labels:
```

```
 19 Exp: 0] 40 ms, ret code 8 L 2 10.3.4.4 10.4.5.5 MRU 1504 [Labels: implicit-null Exp:
0] 32 ms, ret code 8 ! 3 10.4.5.5 40 ms, ret code 3
Router# ping mpls ipv4 10.5.5.5/32

Sending 5, 100-byte MPLS Echos to 10.5.5.5/32
     timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
!    size 100, reply addr 10.4.5.5, return code 3
!    size 100, reply addr 10.4.5.5, return code 3
!    size 100, reply addr 10.4.5.5, return code 3
!    size 100, reply addr 10.4.5.5, return code 3
!    size 100, reply addr 10.4.5.5, return code 3
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/31/32 ms
```

Enabling LSP Ping to Detect LSP Breakages Caused by Untagged Interfaces Example

The following example shows the extra label that is added to the end of the label stack when there is explicit-null label shimming:

```
Switch# trace mpls ipv4 10.131.159.252/32 force-explicit-null

Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes:
  '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.191.252 MRU 1492 [Labels: 16/18/explicit-null Exp: 0/0/0]
L 1 10.131.191.229 MRU 1508 [Labels: 18/explicit-null Exp: 0/0] 0 ms
L 2 10.131.159.225 MRU 1508 [Labels: explicit-null Exp: 0] 0 ms
! 3 10.131.159.234 4 ms
```

The following example shows the command output when there is not explicit-null label shimming:

```
Switch# trace mpls ipv4 10.131.159.252/32

Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 10.131.191.252 MRU 1496 [Labels: 16/18 Exp: 0/0]
L 1 10.131.191.229 MRU 1508 [Labels: 18 Exp: 0] 4 ms
L 2 10.131.159.225 MRU 1504 [Labels: implicit-null Exp: 0] 4 ms
! 3 10.131.159.234 4 ms
```

Viewing the AToM VCCV Capabilities Advertised to and Received from the Peer Example

The following example shows that router PE1 advertises both AToM VCCV Type 1 and Type 2 switching capabilities and that the remote router PE2 advertises only a Type 2 switching capability.

```
Router# show mpls l2transport binding

  Destination Address: 10.131.191.252,  VC ID: 333
    Local Label:  16
        Cbit: 1,    VC Type: Ethernet,    GroupID: 0
        MTU: 1500,   Interface Desc: n/a
        VCCV Capabilities: Type 1, Type 2 <----- Locally advertised VCCV capabilities
    Remote Label: 19
        Cbit: 1,    VC Type: Ethernet,    GroupID: 0
        MTU: 1500,   Interface Desc: n/a
        VCCV Capabilities: Type 2           <-----Remotely advertised VCCV capabilities
```

Additional References

Related Documents

Related Topic	Document Title
Usage examples for the IP ping and IP traceroute commands	Understanding the Ping and Traceroute Commands
Configuration and verification tasks for MPLS LDP	MPLS Label Distribution Protocol (LDP) Overview
Configuration and verification tasks for AToM	Any Transport over MPLS
Switching services commands	*Multiprotocol Label Switching Command Reference*
Automatic detection of which PE routers are added to or removed from the Virtual Private LAN Service (VPLS) domain	Information About VPLS Autodiscovery: BGP Based

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
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use the Cisco MIB Locator, found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
draft-ietf-pwe3-vccv-01.txt	Pseudo-Wire (PW) Virtual Circuit Connection Verification (VCCV)
RFC 2113	IP Router Alert Option
RFC 4379	Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures

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 LSP Ping Traceroute for LDP TE and LSP Ping for VCCV

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 7: Feature Information for MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV

Feature Name	Releases	Feature Information
MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV	12.0(27)S 12.2(18)SXE 12.4(6)T 12.2(28)SB 12.0(32)SY 12.4(11)T 12.2(31)SB2 12.2(33)SRB 12.2(33)SXH 12.3(33)SXI	The MPLS LSP Ping/Traceroute for LDP/TE, and LSP Ping for VCCV feature helps service providers monitor label switched paths and quickly isolate MPLS forwarding problems. In Cisco IOS Release 12.0(27)S, this feature was introduced. The following commands were introduced: **ping mpls** and **trace mpls**. The feature was incorporated into Cisco IOS Release 12.2(18)SXE. The following commands were modified: **ping mpls** and **trace mpls**. In Cisco IOS Release 12.4(6)T, the **mpls oam** command was introduced and the **trace mpls** command was modified. This feature was integrated into Cisco IOS Release 12.2(28)SB. The feature was incorporated into Cisco IOS Release 12.0(32)SY. The **show mpls oam echo statistics** command was added. The feature was incorporated into Cisco IOS Release 12.4(11)T. AToM Virtual Circuit Connection Verification (VCCV) is supported. The following commands were modified: **mpls oam**, **ping mpls**, and **trace mpls**. The feature was incorporated into Cisco IOS Release 12.2(31)SB2. In Cisco IOS Release 12.2(33)SRB, support for FEC 129 was added. This feature was integrated into Cisco IOS Release 12.2(33)SXH. This feature was integrated into Cisco IOS Release 12.2(33)SXI.

Glossary

FEC --forwarding equivalence class. A set of packets that can be handled equivalently for forwarding purposes and are thus suitable for binding to a single label. Examples include the set of packets destined for one address prefix and the packets in any flow.

flow --A set of packets traveling between a pair of hosts, or between a pair of transport protocol ports on a pair of hosts. For example, packets with the same source address, source port, destination address, and destination port might be considered a flow.

A flow is also a stream of data traveling between two endpoints across a network (for example, from one LAN station to another). Multiple flows can be transmitted on a single circuit.

fragmentation --The process of breaking a packet into smaller units when they are to be transmitted over a network medium that cannot support the original size of the packet.

ICMP -- Internet Control Message Protocol. A network layer Internet protocol that reports errors and provides other information relevant to IP packet processing. It is documented in RFC 792.

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.

localhost --A name that represents the host router (device). The localhost uses the reserved loopback IP address 127.0.0.1.

LSP --label switched path. A connection between two routers in which MPLS forwards the packets.

LSPV --Label Switched Path Verification. An LSP Ping subprocess. It encodes and decodes MPLS echo requests and replies, and it interfaces with IP, MPLS, and AToM switching for sending and receiving MPLS echo requests and replies. At the MPLS echo request originator router, LSPV maintains a database of outstanding echo requests for which echo responses have not been received.

MPLS router alert label--An MPLS label of 1. An MPLS packet with a router alert label is redirected by the router to the Route Processor (RP) processing level for handling. This allows these packets to bypass any forwarding failures in hardware routing tables.

MRU --maximum receive unit. Maximum size, in bytes, of a labeled packet that can be forwarded through an LSP.

MTU --maximum transmission unit. Maximum packet size, in bytes, that a particular interface can send or receive.

punt --Redirect packets with a router alert from the line card or interface to Route Processor (RP) level processing for handling.

PW --pseudowire. A form of tunnel that carries the essential elements of an emulated circuit from one provider edge (PE) router to another PE router over a packet-switched network.

RP --Route Processor. The processor module in a Cisco 7000 series router that contains the CPU, system software, and most of the memory components that are used in the router. It is sometimes called a supervisory processor.

RSVP --Resource Reservation Protocol. A protocol that supports the reservation of resources across an IP network. Applications running on IP end systems can use RSVP to indicate to other nodes the nature (bandwidth, jitter, maximum burst, and so on) of the packet streams they want to receive. RSVP depends on IPv6. Is is also known as Resource Reservation Setup Protocol.

TLV --type, length, values. A block of information included in a Cisco Discovery Protocol address.

TTL hiding--Time-to-live is a parameter you can set that indicates the maximum number of hops a packet should take to reach its destination.

UDP --User Datagram Protocol. Connectionless transport layer protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without acknowledgments or guaranteed delivery, so error processing and retransmission must be handled by other protocols. UDP is defined in RFC 768.

CHAPTER 2

MPLS LSP Ping, Traceroute, and AToM VCCV

As Multiprotocol Label Switching (MPLS) deployments increase and the traffic types they carry increase, the ability of service providers to monitor label switched paths (LSPs) and quickly isolate MPLS forwarding problems is critical to their ability to offer services. The MPLS LSP Ping, Traceroute, and AToM VCCV feature helps them mitigate these challenges.

The MPLS LSP Ping, Traceroute, and AToM VCCV feature can detect when an LSP fails to deliver user traffic.

- You can use MPLS LSP Ping to test LSP connectivity for IPv4 Label Distribution Protocol (LDP) prefixes, traffic engineering (TE) Forwarding Equivalence Classes (FECs), and AToM FECs.
- You can use MPLS LSP Traceroute to trace the LSPs for IPv4 LDP prefixes and TE tunnel FECs.
- Any Transport over MPLS Virtual Circuit Connection Verification (AToM VCCV) allows you to use MPLS LSP Ping to test the pseudowire (PW) section of an AToM virtual circuit (VC).

Internet Control Message Protocol (ICMP) ping and trace are often used to help diagnose the root cause when a forwarding failure occurs. The MPLS LSP Ping, Traceroute, and AToM VCCV feature extends this diagnostic and troubleshooting ability to the MPLS network and aids in the identification of inconsistencies between the IP and MPLS forwarding tables, inconsistencies in the MPLS control and data plane, and problems with the reply path.

The MPLS LSP Ping, Traceroute, and AToM VCCV feature uses MPLS echo request and reply packets to test LSPs. The Cisco implementation of MPLS echo request and echo reply are based on the Internet Engineering Task Force (IETF) Internet-Draft *Detecting MPLS Data Plane Failures*.

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 LSP Ping, Traceroute, and AToM VCCV

Before you use the MPLS LSP Ping, Traceroute, and AToM VCCV feature, you should:

- Determine the baseline behavior of your Multiprotocol Label Switching (MPLS) network. For example:
    - What is the expected MPLS experimental (EXP) treatment?
    - What is the expected maximum size packet or maximum transmission unit (MTU) of the label switched path?
    - What is the topology? What are the expected label switched paths? How many links in the label switching path (LSP)? Trace the paths of the label switched packets including the paths for load balancing.
- Understand how to use MPLS and MPLS applications, including traffic engineering, Any Transport over MPLS (AToM), and Label Distribution Protocol (LDP). You need to
    - Know how LDP is configured
    - Understand AToM concepts
    - Be able to troubleshoot a TE tunnel
- Understand label switching, forwarding, and load balancing.

Restrictions for MPLS LSP Ping, Traceroute, and AToM VCCV

- You cannot use MPLS LSP Traceroute to trace the path taken by Any Transport over Multiprotocol Label Switching (AToM) packets. MPLS LSP Traceroute is not supported for AToM. (MPLS LSP Ping is supported for AToM.) However, you can use MPLS LSP Traceroute to troubleshoot the Interior Gateway Protocol (IGP) LSP that is used by AToM.
- You cannot use MPLS LSP Ping or Traceroute to validate or trace MPLS Virtual Private Networks (VPNs).
- You cannot use MPLS LSP Traceroute to troubleshoot label switching paths (LSPs) that employ time-to-live (TTL) hiding.

Information About MPLS LSP Ping, Traceroute, and AToM VCCV

MPLS LSP Ping Operation

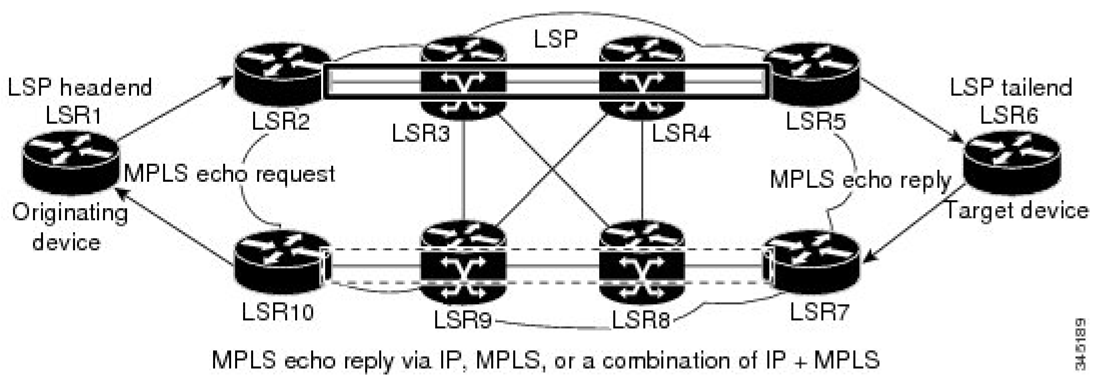
MPLS LSP Ping uses Multiprotocol Label Switching (MPLS) echo request and reply packets to validate a label switched path (LSP). Both an MPLS echo request and an MPLS echo reply are User Datagram Protocol (UDP) packets with source and destination ports set to 3503.

The MPLS echo request packet is sent to a target device through the use of the appropriate label stack associated with the LSP to be validated. Use of the label stack causes the packet to be switched inband of the LSP (that is, forwarded over the LSP itself). The destination IP address of the MPLS echo request packet is different from the address used to select the label stack. The destination address of the UDP packet is defined as a 127.*x* .*y* .*z* /8 address. This prevents the IP packet from being IP switched to its destination if the LSP is broken.

An MPLS echo reply is sent in response to an MPLS echo request. It is sent as an IP packet and forwarded using IP, MPLS, or a combination of both types of switching. The source address of the MPLS echo reply packet is an address from the device generating the echo reply. The destination address is the source address of the device in the MPLS echo request packet.

The figure below shows the echo request and echo reply paths for MPLS LSP Ping.

Figure 7: MPLS LSP Ping Echo Request and Echo Reply Paths



If you initiate an MPLS LSP Ping request at LSR1 to a Forwarding Equivalence Class (FEC), at LSR6, you get the results shown in the table below .

Table 8: MPLS LSP Ping Example

Step	Device	Action
1	LSR1	Initiates an MPLS LSP Ping request for an FEC at the target device LSR6 and sends an MPLS echo request to LSR2.

Step	Device	Action
1	LSR2	Receives and forwards the MPLS echo request packet through transit devices LSR3 and LSR4 to the penultimate device LSR5.
1	LSR5	Receives the MPLS echo request, pops the MPLS label, and forwards the packet to LSR6 as an IP packet.
1	LSR6	Receives the IP packet, processes the MPLS echo request, and sends an MPLS echo reply to LSR1 through an alternate route.
1	LSR7 to LSR10	Receive and forward the MPLS echo reply back toward LSR1, the originating device.
1	LSR1	Receives the MPLS echo reply in response to the MPLS echo request.

You can use MPLS LSP Ping to validate IPv4 Label Distribution Protocol (LDP), Any Transport over MPLS (AToM), and IPv4 Resource Reservation Protocol (RSVP) FECs by using appropriate keywords and arguments with the command:

```
ping mpls
 {ipv4

destination-address destination-mask
 | pseudowire

ipv4-address
 vc-id

| traffic-eng

tunnel-interface tunnel-number
}
```

MPLS LSP Traceroute Operation

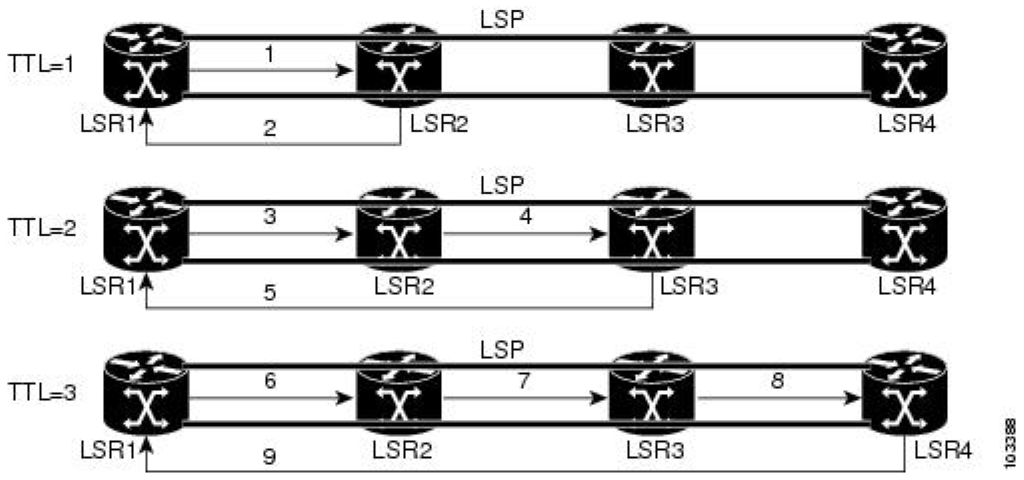
MPLS LSP Traceroute also uses Multiprotocol Label Switching (MPLS) echo request and reply packets to validate a label switched path (LSP). The echo request and echo reply are User Datagram Protocol (UDP) packets with source and destination ports set to 3503.

The MPLS LSP Traceroute feature uses time-to-live (TTL) settings to force expiration of the TTL along an LSP. MPLS LSP Traceroute incrementally increases the TTL value in its MPLS echo requests (TTL = 1, 2, 3, 4, ...) to discover the downstream mapping of each successive hop. The success of the LSP traceroute depends on the transit device processing the MPLS echo request when it receives a labeled packet with a TTL of 1. On Cisco devices, when the TTL expires, the packet is sent to the Route Processor (RP) for processing.

The transit device returns an MPLS echo reply containing information about the transit hop in response to the TTL-expired MPLS packet.

The figure below shows an MPLS LSP Traceroute example with an LSP from LSR1 to LSR4.

Figure 8: MPLS LSP Traceroute Example



If you enter an LSP traceroute to a Forwarding Equivalence Class (FEC) at LSR4 from LSR1, you get the results shown in the table below.

Table 9: MPLS LSP Traceroute Example

Step	**Device**	**MPLS Packet Type and Description**	**Device Action**
1	LSR1	MPLS echo request—With a target FEC pointing to LSR4 and to a downstream mapping.	• Sets the TTL of the label stack to 1. • Sends the request to LSR2.
1	LSR2	MPLS echo reply.	Receives packet with TTL = 1. • Processes the UDP packet as an MPLS echo request. • Finds a downstream mapping, replies to LSR1 with its own downstream mapping based on the incoming label, and sends a reply.

Step	Device	MPLS Packet Type and Description	Device Action
1	LSR1	MPLS echo request—With the same target FEC and the downstream mapping received in the echo reply from LSR2.	• Sets the TTL of the label stack to 2. • Sends the request to LSR2.
1	LSR2	MPLS echo request.	Receives packet with TTL = 2. • Decrements the TTL. • Forwards the echo request to LSR3.
1	LSR3	MPLS reply packet.	Receives packet with TTL = 1. • Processes the UDP packet as an MPLS echo request. • Finds a downstream mapping and replies to LSR1 with its own downstream mapping based on the incoming label.
1	LSR1	MPLS echo request—With the same target FEC and the downstream mapping received in the echo reply from LSR3.	• Sets the TTL of the packet to 3. • Sends the request to LSR2.
1	LSR2	MPLS echo request.	Receives packet with TTL = 3. • Decrements the TTL. • Forwards the echo request to LSR3.
1	LSR3	MPLS echo request.	Receives packet with TTL = 2 • Decrements the TTL. • Forwards the echo request to LSR4.

Step	Device	MPLS Packet Type and Description	Device Action
1	LSR4	MPLS echo reply.	Receives packet with TTL = 1. • Processes the UDP packet as an MPLS echo request. • Finds a downstream mapping and also finds that the device is the egress device for the target FEC. • Replies to LSR1.

You can use MPLS LSP Traceroute to validate IPv4 Label Distribution Protocol (LDP) and IPv4 RSVP FECs by using appropriate keywords and arguments with the **trace mpls** command:

```
trace mpls ipv4 {destination-address destination-mask | traffic-eng
 tunnel-interface tunnel-number}
```

By default, the TTL is set to 30. Therefore, the traceroute output always contains 30 lines, even if an LSP problem exists. This might mean duplicate entries in the output, should an LSP problem occur. The device address of the last point that the trace reaches is repeated until the output is 30 lines. You can ignore the duplicate entries. The following example shows that the trace encountered an LSP problem at the device that has an IP address of 10.6.1.6:

```
Device# traceroute mpls ipv4 10.6.7.4/32
Tracing MPLS Label Switched Path to 10.6.7.4/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.6.1.14 MRU 4470 [Labels: 22 Exp: 0]
R 1 10.6.1.5 MRU 4470 [Labels: 21 Exp: 0] 2 ms
R 2 10.6.1.6 4 ms                  <------ Router address repeated for 2nd to 30th TTL.
R 3 10.6.1.6 1 ms
R 4 10.6.1.6 1 ms
R 5 10.6.1.6 3 ms
R 6 10.6.1.6 4 ms
R 7 10.6.1.6 1 ms
R 8 10.6.1.6 2 ms
R 9 10.6.1.6 3 ms
R 10 10.6.1.6 4 ms
R 11 10.6.1.6 1 ms
R 12 10.6.1.6 2 ms
R 13 10.6.1.6 4 ms
R 14 10.6.1.6 5 ms
R 15 10.6.1.6 2 ms
R 16 10.6.1.6 3 ms
R 17 10.6.1.6 4 ms
R 18 10.6.1.6 2 ms
R 19 10.6.1.6 3 ms
R 20 10.6.1.6 4 ms
R 21 10.6.1.6 1 ms
R 22 10.6.1.6 2 ms
R 23 10.6.1.6 3 ms
R 24 10.6.1.6 4 ms
R 25 10.6.1.6 1 ms
R 26 10.6.1.6 3 ms
R 27 10.6.1.6 4 ms
```

```
R 28 10.6.1.6 1 ms
R 29 10.6.1.6 2 ms
R 30 10.6.1.6 3 ms                  <------ TTL 30.
```

If you know the maximum number of hops in your network, you can set the TTL to a smaller value with the **trace mpls ttl** *maximum-time-to-live* command. The following example shows the same **traceroute** command as the previous example, except that this time the TTL is set to 5.

```
Device# traceroute mpls ipv4 10.6.7.4/32 ttl 5
Tracing MPLS Label Switched Path to 10.6.7.4/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.6.1.14 MRU 4470 [Labels: 22 Exp: 0]
R 1 10.6.1.5 MRU 4474 [No Label] 3 ms
R 2 10.6.1.6 4 ms                 <------ Router address repeated for 2nd to 5th TTL.
R 3 10.6.1.6 1 ms
R 4 10.6.1.6 3 ms
R 5 10.6.1.6 4 ms
```

Any Transport over MPLS Virtual Circuit Connection Verification

AToM Virtual Circuit Connection Verification (AToM VCCV) allows the sending of control packets inband of an AToM pseudowire (PW) from the originating provider edge (PE) device. The transmission is intercepted at the destination PE device, instead of being forwarded to the customer edge (CE) device. This capability allows you to use MPLS LSP Ping to test the PW section of AToM virtual circuits (VCs).

AToM VCCV consists of the following:

- A signaled component in which the AToM VCCV capabilities are advertised during VC label signaling
- A switching component that causes the AToM VC payload to be treated as a control packet

AToM VCCV Signaling

One of the steps involved in Any Transport over Multiprotocol Label Switching (AToM) virtual circuit (VC) setup is the signaling of VC labels and AToM Virtual Circuit Connection Verification (VCCV) capabilities between AToM VC endpoints. The device uses an optional parameter, defined in the Internet Draft *draft-ieft-pwe3-vccv-01.txt,* to communicate the AToM VCCV disposition capabilities of each endpoint.

The AToM VCCV disposition capabilities are categorized as follows:

- Applications—MPLS LSP Ping and Internet Control Message Protocol (ICMP) Ping are applications that AToM VCCV supports to send packets inband of an AToM PW for control purposes.
- Switching modes—Type 1 and Type 2 are switching modes that AToM VCCV uses for differentiating between control and data traffic.

The table below describes AToM VCCV Type 1 and Type 2 switching modes.

Table 10: Type 1 and Type 2 AToM VCCV Switching Modes

Switching Mode	Description
Type 1	Uses a Protocol ID (PID) field in the AToM control word to identify an AToM VCCV packet.

Switching Mode	Description
Type 2	Uses an MPLS Router Alert Label above the VC label to identify an AToM VCCV packet.

Selection of AToM VCCV Switching Types

Cisco devices always use Type 1 switching, if available, when they send MPLS LSP Ping packets over an Any Transport over Multiprotocol Label Switching (AToM) virtual circuit (VC) control channel. Type 2 switching accommodates those VC types and implementations that do not support or interpret the AToM control word.

The table below shows the AToM Virtual Circuit Connection Verification (VCCV) switching mode advertised and the switching mode selected by the AToM VC.

Table 11: AToM VCCV Switching Mode Advertised and Selected by AToM Virtual Circuit

Type Advertised	Type Selected
AToM VCCV not supported	–
Type 1 AToM VCCV switching	Type 1 AToM VCCV switching
Type 2 AToM VCCV switching	Type 2 AToM VCCV switching
Type 1 and Type 2 AToM VCCV switching	Type 1 AToM VCCV switching

An AToM VC advertises its AToM VCCV disposition capabilities in both directions: that is, from the originating device (PE1) to the destination device (PE2), and from PE2 to PE1.

In some instances, AToM VCs might use different switching types if the two endpoints have different AToM VCCV capabilities. If PE1 supports Type 1 and Type 2 AToM VCCV switching and PE2 supports only Type 2 AToM VCCV switching, there are two consequences:

- LSP ping packets sent from PE1 to PE2 are encapsulated with Type 2 switching.
- LSP ping packets sent from PE2 to PE1 use Type 1 switching.

You can determine the AToM VCCV capabilities advertised to and received from the peer by entering the **show mpls l2transport binding** command at the PE device. For example:

```
Device# show mpls l2transport binding

  Destination Address: 10.131.191.252,  VC ID: 333
    Local Label:  16
        Cbit: 1,    VC Type: FastEthernet,    GroupID: 0
        MTU: 1500,   Interface Desc: n/a
        VCCV Capabilities: Type 1, Type 2
    Remote Label: 19
        Cbit: 1,    VC Type: FastEthernet,    GroupID: 0
        MTU: 1500,   Interface Desc: n/a
        VCCV Capabilities: Type 1
```

Command Options for ping mpls and trace mpls

MPLS LSP Ping and Traceroute command options are specified as keywords and arguments on the **ping mpls** and **trace mpls** commands.

The **ping mpls** command provides the options displayed in the command syntax below:

```
ping mpls ipv4{destination-address/destination-mask [destination address-start
address-end increment] [ttl time-to-live] | pseudowire ipv4-address
vc-id vc-id [destination address-start address-end increment] |
traffic-eng tunnel-interface tunnel-number [ttl time-to-live]} [source
source-address] [repeat count] [timeout seconds][{size
 packet-size} | {sweep minimum maximum size-Increment}] [pad pattern]
[reply mode {ipv4|router-alert}] [interval msec]
[exp exp-bits] [verbose]
```

The **trace mpls** command provides the options displayed in the command syntax below:

```
trace mpls {ipv4 destination-address/destination-mask [destination
address-start [address-end [address-increment]]] | traffic-eng tunnel tunnel-interface-number}
[source source-address] [timeout seconds] [reply mode reply-mode]
[ttl maximum-time-to-live] [exp exp-bits]
```

Selection of FECs for Validation

A label switched path (LSP) is formed by labels. Devices learn labels through the Label Distribution Protocol (LDP), traffic engineering (TE), Any Transport over Multiprotocol Label Switching (AToM), or other MPLS applications. You can use MPLS LSP Ping and Traceroute to validate an LSP used for forwarding traffic for a given Forwarding Equivalence Class (FEC). The table below lists the keywords and arguments for the **ping mpls** and **traceroute mpls** commands that allow the selection of an LSP for validation.

Table 12: Selection of LSPs for Validation

FEC Type	ping mpls Keyword and Argument	traceroute mpls Keyword and Argument
LDP IPv4 prefix	**ipv4** *destination-address destination-mask*	**ipv4** *destination-address destination-mask*
MPLS TE tunnel	**traffic-eng** *tunnel-interface tunnel-number*	**traffic-eng** *tunnel-interface tunnel-number*
AToM VC	**pseudowire** *ipv4-address* **vc-id** *vc-id*	MPLS LSP Traceroute does not support the AToM tunnel LSP type for this release.

Reply Mode Options for MPLS LSP Ping and Traceroute

The reply mode is used to control how the responding device replies to a Multiprotocol Label Switching (MPLS) echo request sent by an MPLS LSP Ping or MPLS LSP Traceroute command. The table below describes the reply mode options.

Table 13: Reply Mode Options for a Responding Device

Option	Description
ipv4	Reply with an IPv4 User Datagram Protocol (UDP) packet (default). This is the most common reply mode selected for use with an MPLS LSP Ping and Traceroute command when you want to periodically poll the integrity of a label switched path (LSP). With this option, you do not have explicit control over whether the packet traverses IP or MPLS hops to reach the originator of the MPLS echo request. If the headend device fails to receive a reply, select the router-alert option, "Reply with an IPv4 UDP packet with a router alert." The responding device sets the IP precedence of the reply packet to 6. You implement this option using the **reply mode ipv4** keywords.
router-alert	Reply with an IPv4 UDP packet with a device alert. This reply mode adds the router alert option to the IP header. This forces the packet to be special handled by the Cisco device at each intermediate hop as it moves back to the destination. This reply mode is more expensive, so use the router-alert option only if you are unable to get a reply with the ipv4 option, "Reply with an IPv4 UDP packet." You implement this option using the **reply mode router-alert** keywords

The reply with an IPv4 UDP packet implies that the device should send an IPv4 UDP packet in reply to an MPLS echo request. If you select the ipv4 reply mode, you do not have explicit control over whether the packet uses IP or MPLS hops to reach the originator of the MPLS echo request. This is the mode that you would normally use to test and verify LSPs.

The reply with an IPv4 UDP packet that contains a device alert forces the packet to go back to the destination and be processed by the Route Processor (RP) process switching at each intermediate hop. This bypasses hardware/line card forwarding table inconsistencies. You should select this option when the originating (headend) devices fail to receive a reply to the MPLS echo request.

You can instruct the replying device to send an echo reply with the IP router alert option by using one of the following commands:

```
ping mpls
 {ipv4 destination-address/destination-mask | pseudowire ipv4-address
 vc-idvc-id | traffic-engtunnel-interface tunnel-number}
reply mode  router-alert
```

or

```
trace mpls
 {ipv4destination-address/destination-mask
 | traffic-eng tunnel-interface tunnel-number
} reply mode router-alert
```

However, the reply with a router alert adds overhead to the process of getting a reply back to the originating device. This method is more expensive to process than a reply without a router alert and should be used only if there are reply failures. That is, the reply with a router alert label should only be used for MPLS LSP Ping or MPLS LSP Traceroute when the originating (headend) device fails to receive a reply to an MPLS echo request.

Packet Handling Along Return Path with an IP MPLS Router Alert

When an IP packet that contains an IP router alert option in its IP header or a Multiprotocol Label Switching (MPLS) packet with a router alert label as its outermost label arrives at a device, the device punts (redirects) the packet to the Route Processor (RP) process level for handling. This allows these packets to bypass the forwarding failures in hardware routing tables. The table below describes how IP and MPLS packets with an IP router alert option are handled by the device switching path processes.

Table 14: Switching Path Process Handling of IP and MPLS Router Alert Packets

Incoming Packet	Normal Switching Action	Process Switching Action	Outgoing Packet
IP packet—Router alert option in IP header	A rRouter alert option in the IP header causes the packet to be punted to the process switching path.	Forwards the packet as is.	IP packet—Router alert option in IP header.
	A router alert option in theIP header causes the packet to be punted to the process switching path.	Adds a router alert as the outermost label and forwards as an MPLS packet.	MPLS packet— Outermost label contains a router alert.
MPLS packet—Outermost label contains a router alert	If the router alert label is the outermost label, the packet is punted to the process switching path.	Removes the outermost router alert label, adds an IP router alert option to the IP header, and forwards as an IP packet.	IP packet—Router alert option in IP header.
	If the router alert label is the outermost label, the packet is punted to the process switching path.	Preserves the outermost router alert label and forwards the MPLS packet.	MPLS packet— Outermost label contains a router alert.

Other MPLS LSP Ping and Traceroute Command Options

The table below describes other MPLS LSP Ping and Traceroute command options that can be specified as keywords or arguments with the **ping mpls** command, or with both the **ping mpls** and **trace mpls** commands. Options available to use only on the **ping mpls** command are indicated as such.

Table 15: Other MPLS LSP Ping and Traceroute and AToM VCCV Options

Option	Description
Datagram size	Size of the packet with the label stack imposed. Specified with the **size** *packet-size* keyword and argument. The default size is 100. For use with the MPLS LSP Ping feature only.
Padding	Padding (the pad time-length-value [TLV]) is used as required to fill the datagram so that the MPLS echo request (User Datagram Protocol [UDP] packet with a label stack) is the size specified. Specify with the **pad** *pattern* keyword and argument. For use with the MPLS LSP Ping feature only.
Sweep size range	Parameter that enables you to send a number of packets of different sizes, ranging from a start size to an end size. This parameter is similar to the Internet Control Message Protocol (ICMP) ping sweep parameter. The lower boundary on the sweep range varies depending on the label switched path (LSP) type. You can specify a sweep size range when you use the **ping mpls** command. Use the **sweep** *minimum maximum size-increment* keyword and arguments. For use with the MPLS LSP Ping feature only.
Repeat count	Number of times to resend the same packet. The default is 5 times. You can specify a repeat count when you use the **ping mpls** command. Use the **repeat** *count* keyword and argument. For use with the MPLS LSP Ping feature only.
MPLS echo request source address	Routable address of the sender. The default address is loopback0. This address is used as the destination address in the Multiprotocol Label Switching (MPLS) echo response. Use the **source** *source-address* keyword and argument. For use with the MPLS LSP Ping and Traceroute features.

Option	Description
UDP destination address	A valid 127/8 address. You have the option to specify a single *x.y.z* or a range of numbers between 0.0.0 and *x.y.z* , where *x.y.z* are numbers between 0 and 255 and correspond to 127.*x.y.z*. Use the **destination** {*address* | *address-start address-end increment*} keyword and arguments. The MPLS echo request destination address in the UDP packet is not used to forward the MPLS packet to the destination device. The label stack that is used to forward the echo request routes the MPLS packet to the destination device. The 127/8 address guarantees that the packets are routed to the localhost (the default loopback address of the device processing the address) if the UDP packet destination address is used for forwarding. In addition, the destination address is used to affect load balancing when the destination address of the IP payload is used for load balancing. For use with IPv4 and Any Transport over MPLS (AToM) Forwarding Equivalence Classes (FECs) with the MPLS LSP Ping feature and with IPv4 FECs with the MPLS LSP Traceroute feature.
Time-to-live (TTL)	A parameter you can set that indicates the maximum number of hops a packet should take to reach its destination. The time-to-live (TTL) field in a packet is decremented by 1 each time it travels through a device. For MPLS LSP Ping, the TTL is a value after which the packet is discarded and an MPLS echo reply is sent back to the originating device. Use the **ttl** *time-to-live* keyword and argument. For MPLS LSP Traceroute, the TTL is a maximum time to live and is used to discover the number of downstream hops to the destination device. MPLS LSP Traceroute incrementally increases the TTL value in its MPLS echo requests (TTL = 1, 2, 3, 4, ...) to accomplish this. Use the **ttl** *time-to-live* keyword and argument.
Timeouts	A parameter you can specify to control the timeout in seconds for an MPLS request packet. The range is from 0 to 3600 seconds. The default is 2. Set with the **timeout** *seconds* keyword and argument. For use with the MPLS LSP Ping and Traceroute features.

Option	Description
Intervals	A parameter you can specify to set the time in milliseconds between successive MPLS echo requests. The default is 0. Set with the **interval** *msec* keyword and argument.
Experimental bits	Three experimental bits in an MPLS header used to specify precedence for the MPLS echo reply. (The bits are commonly called EXP bits.) The range is from 0 to 7, and the default is 0. Specify with the **exp** *exp-bits* keyword and argument. For use with the MPLS LSP Ping and Traceroute features.
Verbose	Option that provides additional information for the MPLS echo reply--source address and return codes. For the MPLS LSP Ping feature, this option is implemented with the **verbose** keyword. For use with the MPLS LSP Ping feature only.

MPLS LSP Ping options described in the table above can be implemented by using the following syntax:

```
ping mpls
{ipv4 destination-address destination-mask [destination address-start address-end increment]

 [ttl time-to-live] | pseudowire ipv4-address
 vc-id vc-id
[destination address-start address-end increment] | traffic-eng tunnel-interface
tunnel-number
[ttl time-to-live]}
[source source-address] [repeat count]
[{size packet-size} | {sweep minimum maximum size-Increment}]
[pad pattern]
[timeout seconds] [intervalmsec]
[exp exp-bits] [verbose
```

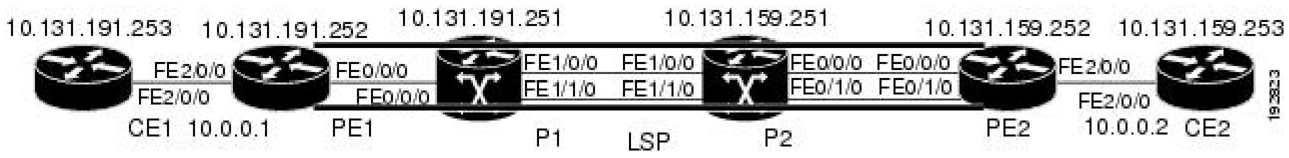
MPLS LSP Traceroute options described in the table below can be implemented by the use of the following syntax:

```
trace mpls
 {ipv4 destination-address destination-mask
 [destination address-start address-end address-increment] | traffic-eng tunnel-interface
tunnel-number}
[source source-address] [timeout seconds]
[ttl maximum-time-to-live]
[exp exp-bits]
```

Option Interactions and Loops

Usage examples for the MPLS LSP Ping and Traceroute and AToM VCCV feature in this and subsequent sections are based on the sample topology shown in the figure below.

Figure 9: Sample Topology for Configuration Examples



The interaction of some MPLS LSP Ping and Traceroute and AToM VCCV options can cause loops. See the following topic for a description of the loops you might encounter with the **ping mpls** and **trace mpls** commands:

Possible Loops with MPLS LSP Ping

With the MPLS LSP Ping feature, loops can occur if you use the repeat count option, the sweep size range option, or the User Datagram Protocol (UDP) destination address range option.

```
ping mpls
 {ipv4 destination-address/destination-mask
[destination address-start address-end increment] | pseudowire ipv4-address
 vc-id vc-id
 [destination address-start address-end increment] |
traffic-eng tunnel-interface tunnel-number}
[repeat count]
[sweep minimum maximum size-increment]
```

Following is an example of how a loop operates if you use the following keywords and arguments on the **ping mpls** command:

```
Device# ping mpls
 ipv4
 10.131.159.251/32 destination 127.0.0.1 127.0.0.1 0.0.0.1 repeat 2
sweep 1450 1475 25
Sending 2, [1450..1500]-byte MPLS Echos to 10.131.159.251/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
Destination address 127.0.0.1
!
!
Destination address 127.0.0.1
!
!
Destination address 127.0.0.1
!
!
Destination address 127.0.0.1
!
!
```

An **mpls ping** command is sent for each packet size range for each destination address until the end address is reached. For this example, the loop continues in the same manner until the destination address, 127.0.0.1, is reached. The sequence continues until the number is reached that you specified with the **repeat** *count* keyword and argument. For this example, the repeat count is 2. The MPLS LSP Ping loop sequence is as follows:

```
repeat  = 1
  destination address 1 (address-start
)
    for (size from sweep
 minimum
 to maximum
, counting by size-increment
)
      send an lsp ping
  destination address 2 (address-start
 +
address-
increment
)
    for (size from sweep
 minimum
 to maximum
, counting by size-increment
)
      send an lsp ping
  destination address 3 (address-start
 +
address-
increment
 +
address-
increment
)
     for (size from sweep
 minimum
 to maximum
, counting by size-increment
)
      send an lsp ping
.
.
.
  until destination address = address-end
.
.
.
until repeat = count
```

Possible Loop with MPLS LSP Traceroute

With the MPLS LSP Traceroute feature, loops can occur if you use the User Datagram Protocol (UDP) destination address range option and the time-to-live option.

```
trace mpls
 {ipv4

destination-address destination-mask
 [destination

address-start
address-end

address-increment
] | traffic-eng
 tunnel-interface

tunnel-number
[ttl
 maximum-
time-to-live
]
```

Here is an example of how a loop operates if you use the following keywords and arguments on the **trace mpls** command:

```
Device# trace mpls
ipv4
 10.131.159.251/32 destination 127.0.0.1 127.0.0.1 1 ttl 5
Tracing MPLS Label Switched Path to 10.131.159.251/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
Destination address 127.0.0.1
  0 10.131.191.230 MRU 1500 [Labels: 19 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
! 2 10.131.159.225 40 ms
Destination address 127.0.0.2
  0 10.131.191.230 MRU 1500 [Labels: 19 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
! 2 10.131.159.225 40 ms
Destination address 127.0.0.3
  0 10.131.191.230 MRU 1500 [Labels: 19 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
! 2 10.131.159.225 48 ms
```

An **mpls trace** command is sent for each TTL from 1 to the maximum TTL (**ttl** *maximum-time-to-live* keyword and argument) for each destination address until the address specified with the destination *end-address* argument is reached. For this example, the maximum TTL is 5 and the end destination address is 127.0.0.1. The MPLS LSP Traceroute loop sequence is as follows:

```
destination address 1 (address-start
)
  for (ttl
 from 1 to maximum-time-to-live
)
    send an lsp trace
destination address 2 (address-start
 + address-increment
)
  for (ttl
 from 1 to maximum-time-to-live
)
    send an lsp trace
destination address 3 (address-start
 + address-increment
 + address-increment
)
  for (ttl
 from 1 to
maximum-time-to-live)
    send an lsp trace
.
.
.
until destination address = address-end
```

MPLS Echo Request Packets Not Forwarded by IP

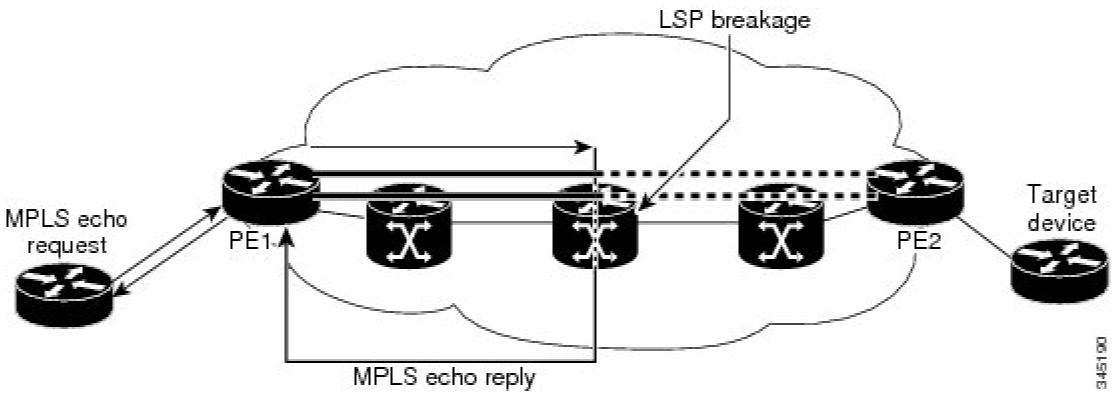
Multiprotocol Label Switching (MPLS) echo request packets sent during a label switched path (LSP) ping are never forwarded by IP. The IP header destination address field in an MPLS echo request packet is a 127.*x.y.z* /8 address. Devices should not forward packets using a 127.*x.y.z* /8 address. The 127.*x.y.z* /8 address corresponds to an address for the local host.

The use of a 127.*x* .*y* .*z* address as a destination address of the User Datagram Protocol (UDP) packet is significant in that the MPLS echo request packet fails to make it to the target device if a transit device does not label switch the LSP. This allows for the detection of LSP breakages.

- If an LSP breakage occurs at a transit device, the MPLS echo packet is not forwarded, but consumed by the device.
- If the LSP is intact, the MPLS echo packet reaches the target device and is processed by the terminal point of the LSP.

The figure below shows the path of the MPLS echo request and reply when a transit device fails to label switch a packet in an LSP.

Figure 10: Path When Transit Device Fails to Label Switch a Packet



Note An Any Transport over MPLS (AToM) payload does not contain usable forwarding information at a transit device because the payload might not be an IP packet. An MPLS virtual private network (VPN) packet, although an IP packet, does not contain usable forwarding information at a transit device because the destination IP address is only significant to the virtual routing and forwarding (VRF) instances at the endpoints of the MPLS network.

Information Provided by the Device Processing LSP Ping or LSP Traceroute

The table below describes the characters that the device processing an LSP ping or LSP traceroute packet returns to the sender about the failure or success of the request.

You can also view the return code for an MPLS LSP Ping operation if you enter the **ping mpls verbose** command.

Table 16: LSP Ping and Traceroute Reply Characters

Character	Meaning
Period "."	A timeout occurs before the target device can reply.
U	The target device is unreachable.

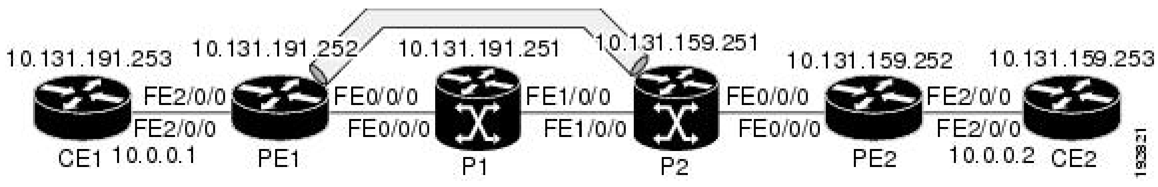
Character	Meaning
R	The device processing the Multiprotocol Label Switching (MPLS) echo request is a downstream device but is not the destination.
Exclamation mark “!”	Replying device is an egress for the destination.
Q	Echo request was not successfully transmitted. This could be returned because of insufficient memory or more probably because no label switched path (LSP) exists that matches the Forwarding Equivalence Class (FEC) information.
C	Replying device rejected the echo request because it was malformed.

MTU Discovery in an LSP

During an MPLS LSP Ping, Multiprotocol Label Switching (MPLS) echo request packets are sent with the IP packet attribute set to do not fragment. That is, the DF bit is set in the IP header of the packet. This allows you to use the MPLS echo request to test for the MTU that can be supported for the packet through the label switched path (LSP) without fragmentation.

The figure below shows a sample network with a single LSP from PE1 to PE2 formed with labels advertised by means of LDP.

Figure 11: Sample Network with LSP—Labels Advertised by LDP



You can determine the maximum receive unit (MRU) at each hop by tracing the LSP using the MPLS Traceroute feature. The MRU is the maximum size of a labeled packet that can be forwarded through an LSP. The following example shows the results of a **trace mpls** command when the LSP is formed with labels created by the Label Distribution Protocol (LDP):

```
Device# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.131.191.230 MRU 1496 [Labels: 22/19 Exp: 0/0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 40 ms
R 2 10.131.159.229 MRU 1504 [implicit-null] 28 ms
! 3 10.131.159.230 40 ms
```

You can determine the MRU for the LSP at each hop through the use of the **show forwarding detail** command:

```
Device# show mpls forwarding 10.131.159.252 detail

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
22     19          10.131.159.252/32 0          Tu1        point2point
        MAC/Encaps=14/22, MRU=1496, Tag Stack{22 19}, via Et0/0
        AABBCC009700AABBCC0098008847 0001600000013000
        No output feature configured
```

To determine the maximum sized echo request that will fit on the LSP, you can find the IP MTU by using the **show interface** *type number* command.

```
Device# show interface e0/0

FastEthernet0/0/0 is up, line protocol is up
  Hardware is Lance, address is aabb.cc00.9800 (bia aabb.cc00.9800)
  Internet address is 10.131.191.230/30
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load ½55
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:01, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     377795 packets input, 33969220 bytes, 0 no buffer
     Received 231137 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     441772 packets output, 40401350 bytes, 0 underruns
     0 output errors, 0 collisions, 10 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

The IP MTU in the **show interface** *type number* example is 1500 bytes. Subtract the number of bytes corresponding to the label stack from the MTU number. From the output of the **show mpls forwarding** command, the Tag stack consists of one label (21). Therefore, the largest MPLS echo request packet that can be sent in the LSP, shown in the figure above, is 1500 - (2 x 4) = 1492.

You can validate this by using the following **ping mpls** command:

```
Device# ping mpls ipv4 10.131.159.252/32 sweep 1492 1500 1 repeat 1
Sending 1, [1492..1500]-byte MPLS Echos to 10.131.159.252/32,
     timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
!QQQQQQQQ
Success rate is 11 percent (1/9), round-trip min/avg/max = 40/40/40 ms
```

In this command, only packets of 1492 bytes are sent successfully, as indicated by the exclamation point (!). Packets of byte sizes 1493 to 1500 are source-quenched, as indicated by the Q.

You can pad an MPLS echo request so that a payload of a given size can be tested. The pad TLV is useful when you use the MPLS echo request to discover the MTU supportable by an LSP. MTU discovery is extremely important for applications like AToM that contain non-IP payloads that cannot be fragmented.

LSP Network Management

To manage a Multiprotocol Label Switching (MPLS) network you must have the ability to monitor label switched paths (LSPs) and quickly isolate MPLS forwarding problems. You need ways to characterize the liveliness of an LSP and reliably detect when a label switched path fails to deliver user traffic.

You can use MPLS LSP Ping to verify the LSP that is used to transport packets destined for IPv4 Label Distribution Protocol (LDP) prefixes, traffic engineering (TE) tunnels, and Any Transport over MPLS pseudowire Forwarding Equivalence Classes (AToM PW FECs). You can use MPLS LSP Traceroute to trace LSPs that are used to carry packets destined for IPv4 LDP prefixes and TE tunnel FECs.

An MPLS echo request is sent through an LSP to validate it. A TTL expiration or LSP breakage causes the transit device to process the echo request before it gets to the intended destination and returns an MPLS echo reply that contains an explanatory reply code to the originator of the echo request.

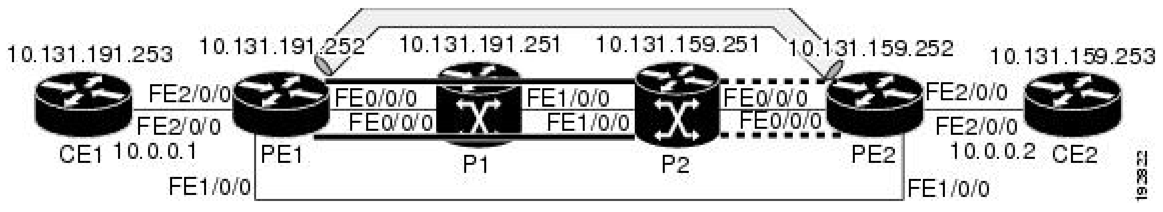
The successful echo request is processed at the egress of the LSP. The echo reply is sent via an IP path, an MPLS path, or a combination of both back to the originator of the echo request.

ICMP ping and trace Commands and Troubleshooting

Internet Control Message Protocol (ICMP) **ping** and **trace** commands are often used to help diagnose the root cause of a failure. When a label switched path (LSP) is broken, the packet might make its way to the target device by way of IP forwarding, thus making ICMP ping and traceroute unreliable for detecting Multiprotocol Label Switching (MPLS) forwarding problems. The MPLS LSP Ping, Traceroute and AToM VCCV feature extends this diagnostic and troubleshooting ability to the MPLS network and handles inconsistencies between the IP and MPLS forwarding tables, inconsistencies in the MPLS control and data plane, and problems with the reply path.

The figure below shows a sample topology with a Label Distribution Protocol (LDP) LSP and traffic engineering (TE) tunnel LSP.

Figure 12: Sample Topology with LDP and TE Tunnel LSPs



This section contains the following topics:

MPLS LSP Ping and Traceroute Discovers LSP Breakage

Configuration for Sample Topology

These are sample topology configurations for the troubleshooting examples in the following sections (see the figure above). There are the six sample device configurations.

Device CE1 Configuration

```
version 12.0
!
hostname ce1
!
enable password lab
!
interface Loopback0
 ip address 10.131.191.253 255.255.255.255
 no ip directed-broadcast
!
interface FastEthernet2/0/0
 ip address 10.0.0.1 255.255.255.255
 no ip directed-broadcast
 no keepalive
 no cdp enable
!
end
```

Device PE1 Configuration

```
version 12.0
!
hostname pe1
!
ip cef
mpls label protocol ldp
mpls traffic-eng tunnels
no mpls traffic-eng auto-bw timers frequency 0
mpls ldp discovery targeted-hello accept
!
interface Loopback0
 ip address 10.131.191.252 255.255.255.255
 no ip directed-broadcast
!
interface Tunnel1
 ip unnumbered Loopback0
 no ip directed-broadcast
 mpls label protocol ldp
 mpls ip
 tunnel destination 10.131.159.255
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 2 2
 tunnel mpls traffic-eng bandwidth  512
 tunnel mpls traffic-eng path-option 1 dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 no ip directed-broadcast
 shutdown
 mpls label protocol ldp
 mpls ip
 tunnel destination 10.131.159.255
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth  100
 tunnel mpls traffic-eng path-option 1 dynamic
!
interface FastEthernet0/0/0
 ip address 10.131.191.230 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface FastEthernet1/0/0
```

```
 ip address 10.131.159.246 255.255.255.255
 no ip directed-broadcast
 no shutdown
 mpls ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface FastEthernet2/0/0
 no ip address
 no ip directed-broadcast
 no cdp enable
 xconnect 10.131.159.252 333 encapsulation mpls
!
interface FastEthernet3/0/0
 no ip address
 no ip directed-broadcast
 shutdown
!
router ospf 1
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.159.244 0.0.0.3 area 0
 network 10.131.191.228 0.0.0.3 area 0
 network 10.131.191.232 0.0.0.3 area 0
 network 10.131.191.252 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip classless

end
```

Device P1 Configuration

```
version 12.0
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname p1
!
enable password lab
!
ip cef
mpls label protocol ldp
mpls ldp logging neighbor-changes
mpls traffic-eng tunnels
no mpls traffic-eng auto-bw timers frequency 0
mpls ldp discovery targeted-hello accept
!
interface Loopback0
 ip address 10.131.191.251 255.255.255.255
 no ip directed-broadcast
!
interface FastEthernet0/0/0
 ip address 10.131.191.229 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface FastEthernet1/0/0
 ip address 10.131.159.226 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
router ospf 1
```

```
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.159.224 0.0.0.3 area 0
 network 10.131.191.228 0.0.0.3 area 0
 network 10.131.191.251 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
end
```

Device P2 Configuration

```
version 12.0
hostname p2
!
ip cef
mpls label protocol ldp
mpls ldp logging neighbor-changes
mpls traffic-eng tunnels
no mpls traffic-eng auto-bw timers frequency 0
mpls ldp discovery directed-hello accept
!
!
interface Loopback0
 ip address 10.131.159.251 255.255.255.255
 no ip directed-broadcast
!
interface FastEthernet0/0/0
 ip address 10.131.159.229 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface FastEthernet1/0/0
 ip address 10.131.159.225 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
router ospf 1
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.159.224 0.0.0.3 area 0
 network 10.131.159.228 0.0.0.3 area 0
 network 10.131.159.251 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
end
```

Device PE2 Configuration

```
version 12.0
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname pe2
!
logging snmp-authfail
enable password lab
!
clock timezone EST -5
ip subnet-zero
ip cef
```

```
no ip domain-lookup
mpls label protocol ldp
mpls ldp logging neighbor-changes
mpls ldp explicit-null
mpls traffic-eng tunnels
no mpls traffic-eng auto-bw timers frequency 0
tag-switching tdp discovery directed-hello accept
frame-relay switching
!
!
interface Loopback0
 ip address 10.131.159.252 255.255.255.255
 no ip directed-broadcast
!
interface Tunnel0
 ip unnumbered Loopback0
 no ip directed-broadcast
 tunnel destination 10.131.191.252
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng path-option 5 explicit name as1pe-long-path
!
interface FastEthernet0/0/0
 ip address 10.131.159.230 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 tag-switching ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface FastEthernet1/0/0
 ip address 10.131.159.245 255.255.255.255
 no ip directed-broadcast
 mpls traffic-eng tunnels
 tag-switching ip
 ip rsvp bandwidth 1500 1500
 ip rsvp signalling dscp 0
!
interface FastEthernet2/0/0
 no ip address
 no ip directed-broadcast
 no cdp enable
 xconnect 10.131.191.252 333 encapsulation mpls
!
interface FastEthernet3/0/0
 no ip address
 no ip directed-broadcast
!
interface Serial4/0/0
 no ip address
 no ip directed-broadcast
 shutdown
!
interface Serial5/0/0
 no ip address
 no ip directed-broadcast
 shutdown
!
router ospf 1
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 log-adjacency-changes
 passive-interface Loopback0
 network 10.131.122.0 0.0.0.3 area 0
 network 10.131.159.228 0.0.0.3 area 0
 network 10.131.159.232 0.0.0.3 area 0
 network 10.131.159.244 0.0.0.3 area 0
 network 10.131.159.252 0.0.0.0 area 0
!
ip classless
!
!
ip explicit-path name as1pe-long-path enable
 next-address 10.131.159.229
 next-address 10.131.159.226
 next-address 10.131.191.230
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
 login
!
end
```

Device CE2 Configuration

```
version 12.0
!
hostname ce2
!
enable password lab
!
interface Loopback0
 ip address 10.131.159.253 255.255.255.255
 no ip directed-broadcast
!
interface FastEthernet2/0/0
 ip address 10.0.0.2 255.255.255.255
 no ip directed-broadcast
 no keepalive
 no cdp enable
!
end
```

Verifying That the LSP Is Set Up Correctly

A **show mpls forwarding-table** command shows that tunnel 1 is in the Multiprotocol Label Switching (MPLS) forwarding table.

```
Device# show mpls forwarding-table 10.131.159.252

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
22     19
     [T] 10.131.159.252/32 0           Tu1
        point2point
[T]     Forwarding through a TSP tunnel.
        View additional tagging info with the 'detail' option
```

A **show mpls traffic-eng tunnels tunnel 1** command entered at PE1 displays information about tunnel 1 and verifies that it is forwarding packets with an out label of 22.

```
Device# show mpls traffic-eng tunnels tunnel 1

Name: PE1_t1                           (Tunnel1) Destination: 10.131.159.251
  Status:
    Admin: up         Oper: up     Path: valid       Signalling: connected
    path option 1, type dynamic (Basis for Setup, path weight 20)
  Config Parameters:
    Bandwidth: 512      kbps (Global)  Priority: 2  2   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:  enabled   LockDown: disabled  Loadshare: 512      bw-based
    auto-bw: disabled
  Active Path Option Parameters:
    State: dynamic path option 1 is active
    BandwidthOverride: disabled  LockDown: disabled  Verbatim: disabled
  InLabel  :  -
  OutLabel : FastEthernet0/0/0, 22
  RSVP Signalling Info:
       Src 10.131.191.252, Dst 10.131.159.251, Tun_Id 1, Tun_Instance 28
    RSVP Path Info:
      My Address: 10.131.191.230
      Explicit Route: 10.131.191.229 10.131.159.226 10.131.159.225 10.131.159.251
      Record   Route:   NONE
      Tspec: ave rate=512 kbits, burst=1000 bytes, peak rate=512 kbits
    RSVP Resv Info:
      Record   Route:   NONE
      Fspec: ave rate=512 kbits, burst=1000 bytes, peak rate=512 kbits
  Shortest Unconstrained Path Info:
    Path Weight: 20 (TE)
    Explicit Route: 10.131.191.230 10.131.191.229 10.131.159.226 10.131.159.225
                    10.131.159.251
  History:
    Tunnel:
      Time since created: 9 days, 14 hours, 12 minutes
      Time since path change: 2 minutes, 18 seconds
    Current LSP:
      Uptime: 2 minutes, 18 seconds
    Prior LSP:
      ID: path option 1 [3]
      Removal Trigger: tunnel shutdown
```

A **trace mpls** command issued at PE1 verifies that packets with 22 as the outermost label and 19 as the end of stack label are forwarded from PE1 to PE2.

```
Device# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.131.191.230 MRU 1496 [Labels: 22/19
 Exp: 0/0]
R 1 10.131.159.226 MRU 1504 [Labels: 19 Exp: 0] 40 ms
R 2 10.131.159.229 MRU 1504 [implicit-null] 28 ms
! 3 10.131.159.230 40 ms
```

The MPLS LSP Traceroute to PE2 is successful, as indicated by the exclamation point (!).

Discovering LSP Breakage

A Label Distribution Protocol (LDP) target-session is established between devices PE1 and P2, as shown in the output of the following **show mpls ldp discovery** command:

```
Device# show mpls ldp discovery

 Local LDP Identifier:
    10.131.191.252:0
    Discovery Sources:
    Interfaces:
        FastEthernet0/0/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
        Tunnel1 (ldp): Targeted -> 10.131.159.251
    Targeted Hellos:
        10.131.191.252 -> 10.131.159.252 (ldp): active/passive, xmit/recv
            LDP Id: 10.131.159.252:0
        10.131.191.252 -> 10.131.159.251 (ldp): active, xmit/recv
            LDP Id: 10.131.159.251:0
```

Enter the following command on the P2 device in global configuration mode:

```
Device# no mpls ldp discovery targeted-hello accept
```

The LDP configuration change causes the targeted LDP session between the headend and tailend of the traffic engineering (TE) tunnel to go down. Labels for IPv4 prefixes learned by P2 are not advertised to PE1. Thus, all IP prefixes reachable by P2 are reachable by PE1 only through IP (not MPLS). In other words, packets destined for those prefixes through Tunnel 1 at PE1 will be IP switched at P2 (which is undesirable).

The following **show mpls ldp discovery** command shows that the LDP targeted-session is down:

```
Device# show mpls ldp discovery

 Local LDP Identifier:
    10.131.191.252:0
    Discovery Sources:
    Interfaces:
       FastEthernet0/0/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
        Tunnel1 (ldp): Targeted -> 10.131.159.251
    Targeted Hellos:
        10.131.191.252 -> 10.131.159.252 (ldp): active/passive, xmit/recv
            LDP Id: 10.131.159.252:0
        10.131.191.252 -> 10.131.159.251 (ldp): active, xmit
```

Enter the **show mpls forwarding-table** command at the PE1 device. The display shows that the outgoing packets are untagged as a result of the LDP configuration changes.

```
Device# show mpls forwarding-table 10.131.159.252

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
22     Untagged[T]
 10.131.159.252/32 0          Tu1        point2point
[T]     Forwarding through a TSP tunnel.
        View additional tagging info with the 'detail' option
```

A **ping mpls** command entered at the PE1 device displays the following:

```
Device# ping mpls ipv4 10.131.159.252/32 repeat 1
Sending 1, 100-byte MPLS Echos to 10.131.159.252/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
R
Success rate is 0 percent (0/1)
```

The **ping mpls** command fails. The R indicates that the sender of the Multiprotocol Label Switching (MPLS) echo reply had a routing entry but no MPLS Forwarding Equivalence Class (FEC) . Entering the **ping mpls verbose** command displays the MPLS label switched path (LSP) echo reply sender address and the return code. You should be able to solve the problem by Telneting to the replying device and inspecting its forwarding and label tables. You might need to look at the neighboring upstream device as well, because the breakage might be on the upstream device.

```
Device# ping mpls ipv4 10.131.159.252/32 repeat 1 verbose
Sending 1, 100-byte MPLS Echos to 10.131.159.252/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
R   10.131.159.225, return code 6
Success rate is 0 percent (0/1)
```

Alternatively, use the LSP **traceroute** command to figure out which device caused the breakage. In the following example, for subsequent values of TTL greater than 2, the same device keeps responding (10.131.159.225). This suggests that the MPLS echo request keeps getting processed by the device regardless of the TTL. Inspection of the label stack shows that P1 pops the last label and forwards the packet to P2 as an IP packet. This explains why the packet keeps getting processed by P2. MPLS echo request packets cannot be forwarded by use of the destination address in the IP header because the address is set to a 127/8 address.

```
Device# trace mpls ipv4 10.131.159.252/32 ttl 5
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.131.191.230 MRU 1500 [Labels: 22 Exp: 0]
R 1 10.131.159.226 MRU 1504 [implicit-null] 40 ms
R 2 10.131.159.225 40 ms
R 3 10.131.159.225 40 ms
R 4 10.131.159.225 40 ms
R 5 10.131.159.225 40 ms
```

MPLS LSP Traceroute Tracks Untagged Cases

This troubleshooting section contains examples of how to use MPLS LSP Traceroute to determine potential issues with packets that are tagged as implicit null and packets that are untagged.

Untagged output interfaces at a penultimate hop do not impact the forwarding of IP packets through a label switched path (LSP) because the forwarding decision is made at the penultimate hop through use of the incoming label. The untagged case causes Any Transport over Multiprotocol Label Switching (AToM) and MPLS virtual private network (VPN) traffic to be dropped at the penultimate hop.

Troubleshooting Implicit Null Cases

In the following example, Tunnel 1 is shut down, and only a label switched path (LSP) formed with Label Distribution Protocol (LDP) labels is established. An implicit null is advertised between the P2 and PE2 devices. Entering an MPLS LSP Traceroute at the PE1 device results in the following display:

```
Device# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.131.191.230 MRU 1500 [Labels: 20 Exp: 0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 80 ms
R 2 10.131.159.229 MRU 1504 [implicit-null] 28 ms
! 3 10.131.159.230 40 ms
```

This output shows that packets are forwarded from P2 to PE2 with an implicit-null label. Address 10.131.159.229 is configured for the P2 Fast Ethernet 0/0/0 out interface for the PE2 device.

Troubleshooting Untagged Cases

Untagged cases are valid configurations for Interior Gateway Protocol (IGP) label switched paths (LSPs) that could cause problems for Multiprotocol Label Switching (MPLS) virtual private networks (VPNs).

A **show mpls forwarding-table** command and a **show mpls ldp discovery** command issued at the P2 device show that the Label Distribution Protocol (LDP) is properly set up:

```
Device# show mpls forwarding-table 10.131.159.252

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
19     Pop tag     10.131.159.252/32 0          Et0/0      10.131.159.230
Device# show mpls ldp discovery
```

```
 Local LDP Identifier:
    10.131.159.251:0
    Discovery Sources:
    Interfaces:
        FastEthernet0/0/0 (ldp): xmit/recv
            LDP Id: 10.131.159.252:0
        FastEthernet1/0/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
```

The **show mpls ldp discovery** command output shows that Fast Ethernet 0/0/0, which connects PE2 to P2, is sending and receiving packets.

If a **no mpls ip** command is entered on Fast Ethernet 0/0/0, this could prevent an LDP session between the P2 and PE2 devices from being established. A **show mpls ldp discovery** command entered on the PE device shows that the MPLS LDP session with the PE2 device is down:

```
Device# show mpls ldp discovery

 Local LDP Identifier:
    10.131.159.251:0
    Discovery Sources:
    Interfaces:
        FastEthernet0/0/0 (ldp): xmit
        FastEthernet1/0/0 (ldp): xmit/recv
            LDP Id: 10.131.191.251:0
```

If the MPLS LDP session to PE2 goes down, the LSP to 10.131.159.252 becomes untagged, as shown by the **show mpls forwarding-table** command:

```
Device# show mpls forwarding-table 10.131.159.252

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
19     Untagged
    10.131.159.252/32 864        Et0/0      10.131.159.230
```

Untagged cases would provide an MPLS LSP Traceroute reply with packets tagged with No Label, as shown in the following display:

```
Device# trace mpls ipv4 10.131.159.252/32
Tracing MPLS Label Switched Path to 10.131.159.252/32, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
  0 10.131.191.230 MRU 1500 [Labels: 20 Exp: 0]
R 1 10.131.159.226 MRU 1500 [Labels: 19 Exp: 0] 80 ms
R 2 10.131.159.229 MRU 1504 [No Label] 28 ms
! 3 10.131.159.230 40 ms
```

MPLS LSP Ping and Traceroute Returns a Q

The Q return code always means that the packet could not be transmitted. The problem can be caused by insufficient memory, but it probably results because a label switched path (LSP) could not be found that matches the Forwarding Equivalence Class (FEC), information that was entered on the command line.

The reason that the packet was not forwarded needs to be determined. To do so, look at the Routing Information Base (RIB), the Forwarding Information Base (FIB), the Label Information Base (LIB), and the MPLS Label Forwarding Information Base (LFIB). Lack of an entry for the FEC in any one of these routing/forwarding bases would return a Q.

The table below lists commands that you can use for troubleshooting when the MPLS echo reply returns a Q.

Table 17: Troubleshooting a Q

Database	Command to View Contents
Routing Information Base	**show ip route**
Label Information Base and MPLS Forwarding Information Base	**show mpls forwarding-table detail**

The following example shows a **ping mpls** command where the MPLS echo request is not transmitted, as shown by the returned Qs:

```
Device# ping mpls ipv4 10.0.0.1/32
Sending 5, 100-byte MPLS Echos to 10.0.0.1/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
QQQQQ
Success rate is 0 percent (0/5)
```

A **show mpls forwarding-table** command and a **show ip route** command demonstrate that the address is not in either routing table:

```
Device# show mpls forwarding-table 10.0.0.1

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
Device# show ip route 10.0.0.1

% Subnet not in table
```

The MPLS echo request is not transmitted because the IPv4 address (10.0.0.1) is not found in either the LFIB or the RIB routing table.

Load Balancing for IPv4 LDP LSPs

An Internet Control Message Protocol (ICMP) ping or trace follows one path from the originating device to the target device. Round robin load balancing of IP packets from a source device is used to discover the various output paths to the target IP address.

For MPLS LSP Ping and Traceroute, Cisco devices use the source and destination addresses in the IP header for load balancing when multiple paths exist through the network to a target device. The Cisco implementation of MPLS might check the destination address of an IP payload to accomplish load balancing (this checking depends on the platform).

To check for load balancing paths, you use the 127.*z.y.x* /8 destination address in the **ping mpls ipvr** *ip-address address-mask* **destination** *address-start address-end address-increment* command. The following examples show that different paths are followed to the same destination. This demonstrates that load balancing occurs between the originating device and the target device.

To ensure that the Fast Ethernet interface 1/0/0 on the PE1 device is operational, you enter the following commands on the PE1 device:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# interface fastethernet 1/0/0
Device(config-if)# no shutdown
Device(config-if)# end
*Dec 31 19:14:10.034: %LINK-3-UPDOWN: Interface FastEthernet1/0/0, changed state to up
*Dec 31 19:14:11.054: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0/0,
changed state to upend
PE1#
*Dec 31 19:14:12.574: %SYS-5-CONFIG_I: Configured from console by console
*Dec 31 19:14:19.334: %OSPF-5-ADJCHG: Process 1, Nbr 10.131.159.252 on FastEthernet1/0/0
from LOADING to FULL, Loading Done
PE1#
```

The following **show mpls forwarding-table** command displays the possible outgoing interfaces and next hops for the prefix 10.131.159.251/32:

```
Device# show mpls forwarding-table 10.131.159.251

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
21     19          10.131.159.251/32 0          FE0/0/0 10.131.191.229
       20          10.131.159.251/32 0          FE1/0/0 10.131.159.245
```

The following **ping mpls** command to 10.131.159.251/32 with a destination UDP address of 127.0.0.1 shows that the path selected has a path index of 0:

```
Device# ping mpls ipv4
 10.131.159.251/32 destination
 127.0.0.1 repeat 1
Sending 1, 100-byte MPLS Echos to 10.131.159.251/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 40/40/40 ms
PE1#
*Dec 29 20:42:40.638: LSPV: Echo Request sent on IPV4 LSP, load_index 2,
pathindex 0
, size 100
*Dec 29 20:42:40.638: 46 00 00 64 00 00 40 00 FF 11 9D 03 0A 83 BF FC
*Dec 29 20:42:40.638: 7F 00 00 01 94 04 00 00 0D AF 0D AF 00 4C 14 70
*Dec 29 20:42:40.638: 00 01 00 00 01 02 00 00 1A 00 00 1C 00 00 00 01
*Dec 29 20:42:40.638: C3 9B 10 40 A3 6C 08 D4 00 00 00 00 00 00 00 00
*Dec 29 20:42:40.638: 00 01 00 09 00 01 00 05 0A 83 9F FB 20 00 03 00
*Dec 29 20:42:40.638: 13 01 AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Dec 29 20:42:40.638: AB CD AB CD
*Dec 29 20:42:40.678: LSPV: Echo packet received: src 10.131.159.225,
dst 10.131.191.252, size 74
*Dec 29 20:42:40.678: AA BB CC 00 98 01 AA BB CC 00 FC 01 08 00 45 C0
*Dec 29 20:42:40.678: 00 3C 32 D6 00 00 FD 11 15 37 0A 83 9F E1 0A 83
*Dec 29 20:42:40.678: BF FC 0D AF 0D AF 00 28 D1 85 00 01 00 00 02 02
*Dec 29 20:42:40.678: 03 00 1A 00 00 1C 00 00 00 01 C3 9B 10 40 A3 6C
*Dec 29 20:42:40.678: 08 D4 C3 9B 10 40 66 F5 C3 C8
```

The following **ping mpls** command to 10.131.159.251/32 with a destination UDP address of 127.0.0.1 shows that the path selected has a path index of 1:

```
Device# ping mpls ipv4 10.131.159.251/32 dest 127.0.0.1 repeat 1
Sending 1, 100-byte MPLS Echos to 10.131.159.251/32,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not transmitted,
       '.' - timeout, 'U' - unreachable,
       'R' - downstream router but not target
Type escape sequence to abort.
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 40/40/40 ms
*Dec 29 20:43:09.518: LSPV: Echo Request sent on IPV4 LSP, load_index 13,
pathindex 1
, size 100
*Dec 29 20:43:09.518: 46 00 00 64 00 00 40 00 FF 11 9D 01 0A 83 BF FC
*Dec 29 20:43:09.518: 7F 00 00 03 94 04 00 00 0D AF 0D AF 00 4C 88 58
```

```
*Dec 29 20:43:09.518: 00 01 00 00 01 02 00 00 38 00 00 1D 00 00 00 01
*Dec 29 20:43:09.518: C3 9B 10 5D 84 B3 95 84 00 00 00 00 00 00 00 00
*Dec 29 20:43:09.518: 00 01 00 09 00 01 00 05 0A 83 9F FB 20 00 03 00
*Dec 29 20:43:09.518: 13 01 AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Dec 29 20:43:09.518: AB CD AB CD
*Dec 29 20:43:09.558: LSPV: Echo packet received: src 10.131.159.229,
dst 10.131.191.252, size 74
*Dec 29 20:43:09.558: AA BB CC 00 98 01 AA BB CC 00 FC 01 08 00 45 C0
*Dec 29 20:43:09.558: 00 3C 32 E9 00 00 FD 11 15 20 0A 83 9F E5 0A 83
*Dec 29 20:43:09.558: BF FC 0D AF 0D AF 00 28 D7 57 00 01 00 00 02 02
*Dec 29 20:43:09.558: 03 00 38 00 00 1D 00 00 00 01 C3 9B 10 5D 84 B3
*Dec 29 20:43:09.558: 95 84 C3 9B 10 5D 48 3D 50 78
```

To see the actual path chosen, you use the **debug mpls lspv packet data** command.

Note The hashing algorithm is nondeterministic. Therefore, the selection of the *address-start* , *address-end* , and *address-increment* arguments for the **destination** keyword might not provide the expected results.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Cisco IOS Multiprotocol Label Switching Command Reference
Switching services commands	Cisco IOS IP Switching Command Reference
Concepts and tasks for configuring MPLS VPNs	*MPLS: Layer 3 VPNs Configuration Guide* (part of the *Multiprotocol Label Switching Configuration Guide Library*)

Standards and RFCs

Standards/RFCs	Title
draft-ietf-mpls-lsp-ping-03.txt	*Detecting MPLS Data Plane Failures*
draft-ietf-pwe3-vccv-01.txt	*Pseudo-Wire (PW) Virtual Circuit Connection Verification (VCCV)*
RFC 2113	*IP Router Alert Option*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for MPLS LSP Ping, Traceroute, and AToM VCCV

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 18: Feature Information for MPLS LSP Ping, Traceroute, and AToM VCCV

Feature Name	Releases	Feature Information
MPLS LSP Ping, Traceroute, and AToM VCCV	12.0(27)S 12.2(28)SB 12.2(33)SXH Cisco IOS XE Release 2.3 Cisco IOS XE Release 3.5S	You can use MPLS LSP Ping to test label switched path (LSP) connectivity for IPv4 Label Distribution Protocol (LDP) prefixes, traffic engineering (TE) Forwarding Equivalence Classes (FECs), and Any Transport over MPLS (AToM) FECs. You can use MPLS LSP Traceroute to trace the LSPs for IPv4 LDP prefixes and TE tunnel FECs. AToM VCCV allows you to use MPLS LSP Ping to test the pseudowire (PW) section of an AToM virtual circuit (VC). In Cisco IOS Release 12.2(28)SB, this feature was enhanced to support the Cisco 10000 series router. This feature was integrated into Cisco IOS Release 12.2(33)SXH and Cisco IOS XE Release 2.3. In Cisco IOS XE Release 3.5S, support was added for the Cisco ASR 903 Router. The following commands were introduced or modified: **debug mpls lspv**, **ping mpls**, **trace mpls**.

Glossary

FEC—Forwarding Equivalence Class. A set of packets that can be handled equivalently for forwarding purposes and are thus suitable for binding to a single label. Examples include the set of packets destined for one address prefix and any flow.

flow—Generally, a set of packets traveling between a pair of hosts, or a pair of transport protocol ports on a pair of hosts. For example, packets with the same source address, source port, destination address, and destination port might be considered a flow.

A flow is also a stream of data traveling between two endpoints across a network (for example, from one LAN station to another). Multiple flows can be transmitted on a single circuit.

fragmentation—Process of breaking a packet into smaller units when transmitting over a network medium that cannot support the original size of the packet.

ICMP— Internet Control Message Protocol. A network layer Internet protocol that reports errors and provides other information relevant to IP packet processing. It is documented in RFC 792.

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.

localhost—A name that represents the host name of a device. The localhost uses the reserved loopback IP address 127.0.0.1.

LSP—label switched path. A connection between two devices that uses MPLS to carry the packets.

LSPV—Label Switched Path Verification. An LSP Ping subprocess that encodes and decodes MPLS echo requests and replies; interfaces with IP, MPLS, and AToM switching for sending and receiving MPLS echo requests and replies; and, at the MPLS echo request originator device, maintains a database of outstanding echo requests for which echo responses have not been received.

MPLS router alert label—An MPLS label of 1. An MPLS packet with a router alert label is redirected by the device to the Route Processor (PR) processing level for handling. This allows these packets to bypass any forwarding failures in hardware routing tables.

MRU—maximum receive unit. Maximum size, in bytes, of a labeled packet that can be forwarded through an LSP.

MTU—maximum transmission unit. Maximum packet size, in bytes, that a particular interface can handle.

punt—Redirect packets with a router alert from the line card or interface to Route Processor (RP) level processing for handling.

PW—pseudowire. A mechanism that carries the essential elements of an emulated circuit from one provider edge (PE) device to another PE device over a packet-switched network.

RP—Route Processor. Processor module in the Cisco 7000 series routers that contains the CPU, system software, and most of the memory components that are used in the device. It is sometimes called a supervisory processor.

RSVP—Resource Reservation Protocol. A protocol that supports the reservation of resources across an IP network. Applications running on IP end systems can use RSVP to indicate to other nodes the nature (bandwidth, jitter, maximum burst, and so on) of the packet streams they want to receive. RSVP depends on IPv6. Is is also known as Resource Reservation Setup Protocol.

UDP—User Datagram Protocol. Connectionless transport layer protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without acknowledgments or guaranteed delivery, requiring that error processing and retransmission be handled by other protocols. UDP is defined in RFC 768.

CHAPTER 3

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 102.

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.

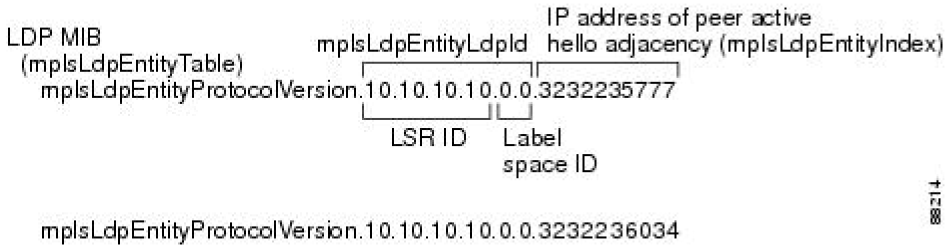
The figure below shows the following indexing:

- 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.

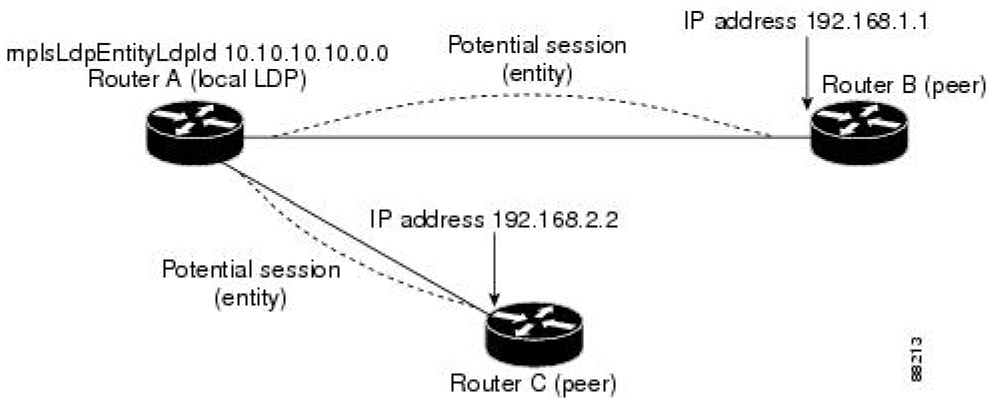
Figure 13: Sample Indexing for an LDP Entity


LDP MIB
(mplsLdpEntityTable)
mplsLdpEntityLdpId
IP address of peer active hello adjacency (mplsLdpEntityIndex)
mplsLdpEntityProtocolVersion.10.10.10.10.0.0.3232235777
LSR ID
Label space ID
mplsLdpEntityProtocolVersion.10.10.10.10.0.0.3232236034
88214

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 14: 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.

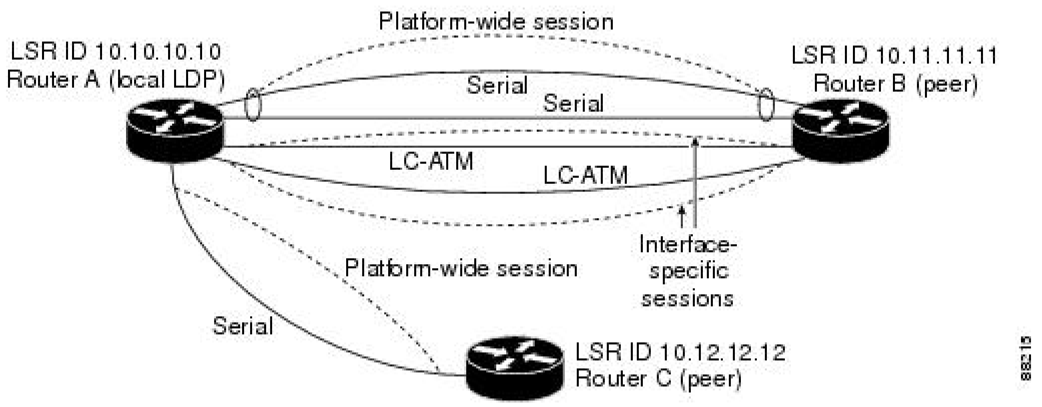
LDP supports the following types of sessions:

- 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 15: 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:

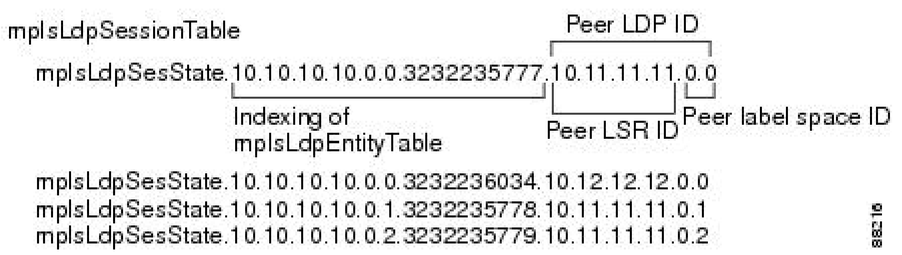
- Peer LDP ID = 10.11.11.11.0.0

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 16: Sample Indexing for an LDP Session



```
mplsLdpSessionTable
  mplsLdpSesState.10.10.10.10.0.0.3232235777.10.11.11.11.0.0

  mplsLdpSesState.10.10.10.10.0.0.3232236034.10.12.12.12.0.0
  mplsLdpSesState.10.10.10.10.0.1.3232235778.10.11.11.11.0.1
  mplsLdpSesState.10.10.10.10.0.2.3232235779.10.11.11.11.0.2
```

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.

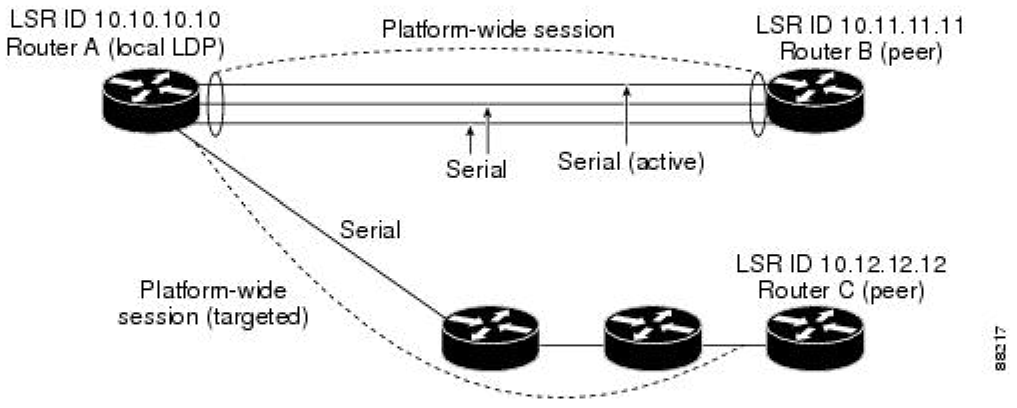
An LDP hello adjacency exists for each link on which LDP runs. Multiple LDP hello adjacencies exist whenever there is more than one link in a session between a router and its peer, such as in a platform-wide session.

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.

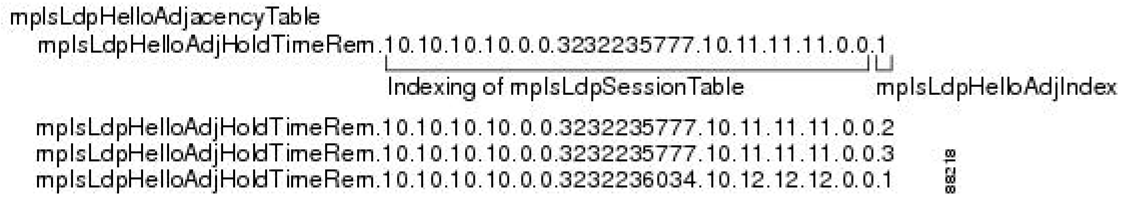
Figure 17: Hello Adjacency



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 18: Sample Indexing for an LDP Hello Adjacency



```
mplsLdpHelloAdjacencyTable
  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232235777.10.11.11.11.0.0.1

  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232235777.10.11.11.11.0.0.2
  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232235777.10.11.11.11.0.0.3
  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232236034.10.12.12.12.0.0.1
```
Indexing of mplsLdpSessionTable
mplsLdpHelloAdjIndex
88218

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 19: 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 20: 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 21: 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 22: 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 23: 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 24: 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 25: 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 26: 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 27: 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 28: 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 29: 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.

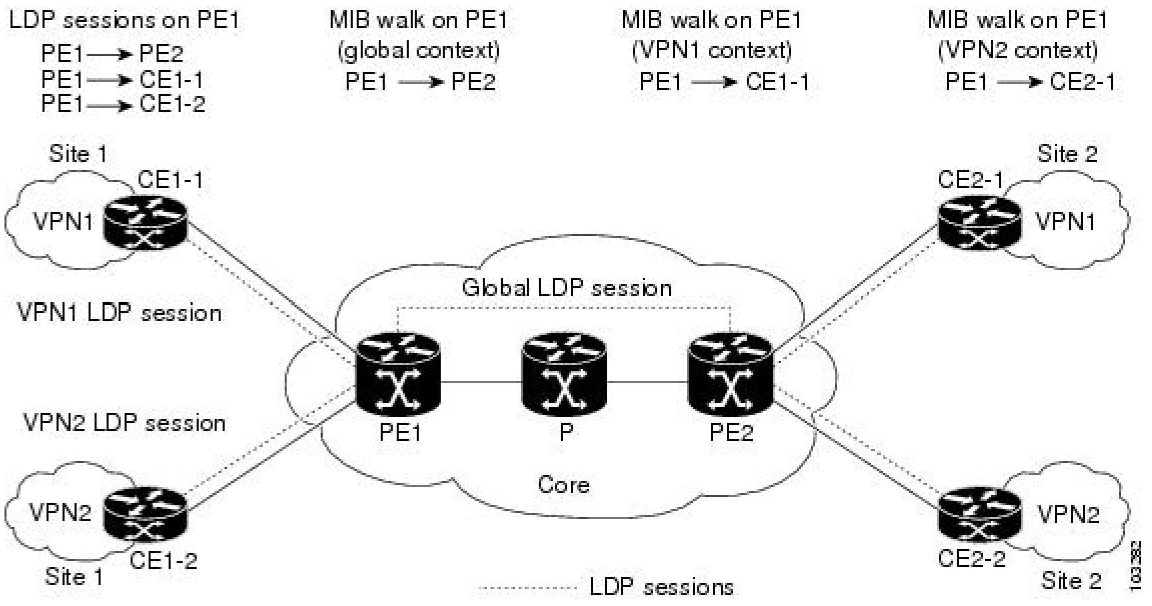
VPN Aware LDP MIB Sessions

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 19: 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:

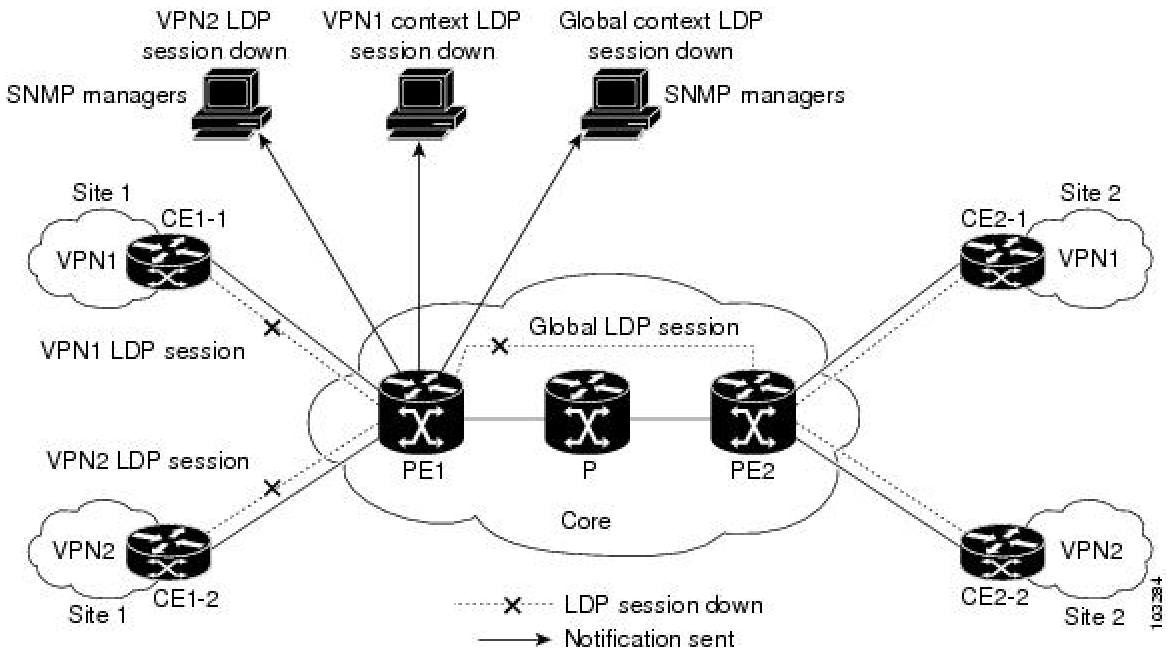
```
Router(config)# snmp-server host host-address trapscommunity
 mpls-ldp
Router(config)# snmp-server enable traps mpls rfc ldp
```

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 20: 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 30: 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 31: 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 32: 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 33: 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 34: 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 35: 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 36: 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 37: 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 140.

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 143.

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 38: 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 15.2(1)SY	

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 . In 15.2(1)SY, the feature was integrated into Cisco IOS Release 15.2(2)SY.

Feature Name	Releases	Feature Information
		The following command was introduced in this feature: **snmp-server enable traps mpls 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.

CHAPTER 4

MPLS Enhancements to Interfaces MIB

This document describes the Multiprotocol Label Switching (MPLS) enhancements to the existing Interfaces MIB (RFC 2233) to support an MPLS layer. This layer provides counters and statistics specifically for MPLS.

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 Enhancements to Interfaces MIB

- Simple Network Management Protocol (SNMP) must be installed and enabled on the label switching routers (LSRs)
- MPLS must be enabled on the LSRs

- MPLS IP must be enabled on an interface or an MPLS traffic engineering (TE) tunnel enabled on an interface

Restrictions for MPLS Enhancements to Interfaces MIB

- Link up and link down traps for the MPLS layer are not supported in this release.
- Write capability using the SNMP SET command is not supported for the MPLS layer in this release.
- Some counters, including discard and multicast, increment on the underlying physical layer; therefore, they equal 0 because they never reach the MPLS layer.
- The high-capacity counters for the MPLS layer interfaces of the Interfaces MIB contain 64 bits of counter data. In previous versions, the high capacity counters displayed 32 bits of counter data.

The following MIB objects are affected:

- - ifHCInOctets
  - ifHCOutOctets
  - ifHCInUcastPkts
  - ifHCOutUcastPkts

When the 64-bit values are less than the value of 232, the 32-bit and 64-bit values are identical.

After the counter increases to more than 232, the counters are different; the 64-bit value is computed by the following formula:

X * (232) + Y

where:

- - X is the number of times the 32-bit counter has rolled.
  - Y is the residual value of the counter after the roll occurred. The Y value equals the 32-bit value.

When the high-capacity counter values are compared to their 32-bit values, there is a period of time that the counter values are not equal. The 64-bit values lag the 32-bit values when the counters poll the 32-bit hardware counters and computing the correct counter value. During the polling and computation interval, the following high-capacity counter values counters might be inconsistent:

- - ifInOctets
  - ifOutOctets
  - ifInUcastPkts
  - ifOutUcastPkts

The inconsistent values can occur if traffic is constantly flowing over an interface and a MIB walk is performed. The 32-bit value is correct at that moment. The 64-bit value lags slightly, because of the polling computations needed to generate it. Once traffic stops flowing over the interface, and a polling period has passed, the two counters are identical and correct.

The lag time depends on the following factors:

- The polling interval used by the Interfaces MIB. The less time the polling interval takes, the more accurate the value is.
- The size of the Interfaces MIB. A large MIB takes a long time to walk and might affect the values found at that instant.
- The number of computations needed to generate the 64-bit value. The number of MPLS-enabled interfaces increases the number of 64-bit counter values that need to be computed.

Information About MPLS Enhancements to Interfaces MIB

Feature Design of the MPLS Enhancements to Interfaces MIB

The Interfaces MIB (IF MIB) provides an SNMP-based method for managing interfaces. Each entry in the IF MIB establishes indexing, statistics, and stacking relationships among underlying physical interfaces, subinterfaces, and Layer 2 protocols that exist within Cisco software.

The enhancements add an MPLS layer to the IF MIB as a Layer 2 protocol to provide statistics for traffic encapsulated as MPLS on an interface. In this structure, MPLS-specific data such as MPLS-encapsulated traffic counters and the MPLS maximum transmission unit (MTU) resides on top of the underlying physical or virtual interface to allow separation from non-MPLS data.

The enhancements also allow you to display indexing, statistics, and stacking relationships using the ifStackTable. MPLS layer interfaces are stacked above the underlying physical or virtual interface that is actually forwarding the MPLS traffic. MPLS traffic engineering tunnels are then stacked above those MPLS layers.

The IF MIB supports several types of interfaces. A virtual interface that provides protocol statistics for MPLS-encapsulated traffic has been added. This interface is stacked above real Cisco interfaces or subinterfaces, such as Fast Ethernet (fe0/1/0) or ATM (at1/1.1).

Cisco software creates a corresponding MPLS layer above each interface capable of supporting MPLS when the MPLS encapsulation is enabled by issuing the **mpls ip** command in interface configuration mode.

You can also create the interface layer if you enable MPLS TE by using the **mpls traffic-eng tunnels** command in interface configuration mode.

Note You must also issue these commands in global configuration mode for MPLS IP or MPLS TE to be enabled.

An IF MIB entry is created when you enable either MPLS IP or MPLS TE tunnels on an interface; the entry is removed when you disable both MPLS IP and MPLS TE.

ifStackTable Objects

The table below defines the ifStackTable objects.

Table 39: ifStackTable Objects and Definitions

Object	Definition
ifStackHigherLayer	The value of ifIndex corresponding to the higher sublayer of the relationship; that is, the sublayer that runs on top of the sublayer identified by the corresponding instance of the ifStackLowerLayer. **Note** Index objects are not accessible in a MIB walk. This value is part of the object identifier (OID) for every object in the ifStackTable.
ifStackLowerLayer	The value of ifIndex corresponding to the lower sublayer of the relationship; that is, the sublayer that runs below the sublayer identified by the corresponding instance of the ifStackHigherLayer. **Note** Index objects are not accessible in a MIB walk. This value is part of the OID for every object in the ifStackTable.
ifStackStatus	Used to create and delete rows in the ifStackTable; status is always active(1) for MPLS.

ifRcvAddressTable Objects

The table below defines the ifRcvAddressTable objects.

Note Entries for the MPLS layer do not appear in the ifRcvAddressTable.

Table 40: ifRcvAddressTable Objects and Descriptions

Object	Definition
ifRcvAddressAddress	An address for which the system accepts packets and frames on this entry's interface. **Note** Index objects are not accessible in a MIB walk. This value is part of the OID for every object in the ifRcvAddressTable.
ifRcvAddressStatus	Used to create and delete rows in the ifRcvAddressTable.
ifRcvAddressType	Type of storage used for each entry in the ifRcvAddressTable.

Interfaces MIB Scalar Objects

The IF MIB supports the following scalar objects:

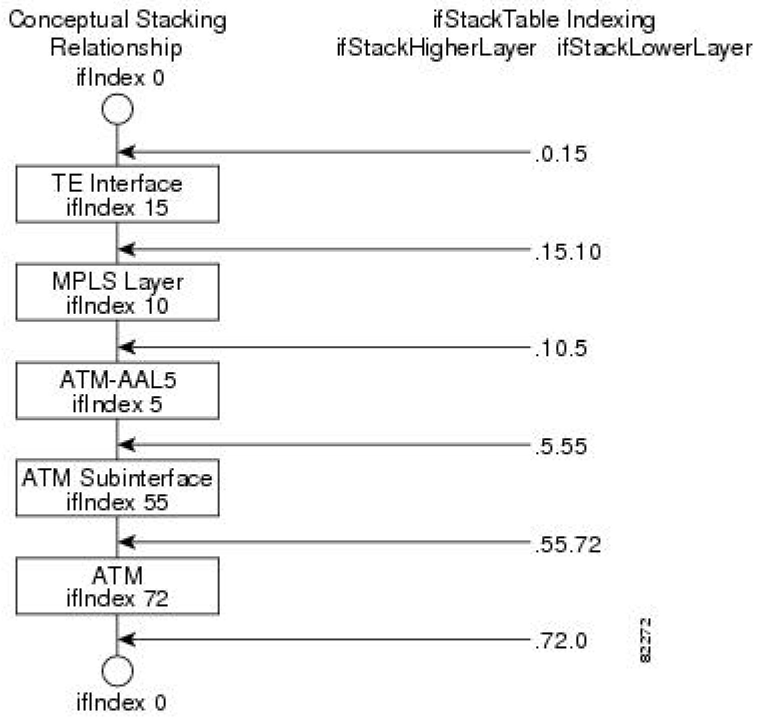
- ifStackLastChange--The value of sysUpTime at the time of the last change of the entire interface stack. A change of the interface stack is defined to be any creation, deletion, or change in value of any instance of ifStackStatus. If the interface stack has been unchanged since the last reinitialization of the local network management subsystem, then this object contains a zero value.
- ifTableLastChange--The value of sysUpTime at the time of the last creation or deletion of an entry in the ifTable. If the number of entries has been unchanged since the last reinitialization of the local network management subsystem, then this object contains a zero value.

Stacking Relationships for MPLS Layer Interfaces

The ifStackTable within the IF MIB provides a conceptual stacking relationship between the interfaces and subinterfaces represented as entries in the ifTable.

The ifStackTable is indexed like a linked list. Each entry shows a relationship between two interfaces providing the ifIndexes of the upper and the lower interface. The entries chain together to show the entire stacking relationship. Each entry links with one another until the stack terminates with an ifIndex of 0 at the highest and lowest ends of the stack. For example, in the figure below, the indexes .10.5 show that ifIndex 10 is stacked upon ifIndex 5. There are 0 entries at the highest and lowest ends of the stack; in the figure, the indexes .0.15 and .72.0 are the highest and lowest ends of the stack, respectively.

Figure 21: Sample ATM Stacking Relationship in the ifStackTable



The table below describes the indexing of the ifStackTable for the layer relationships shown in the figure above.

Note The order of the entries in the table may not be the same as that seen in the MIB walk, which has to follow SNMP ordering rules.

Table 41: Layer Relationships

Layer Relationship (in Descending Order)	ifStackHigherLayer/ifStackLowerLayer
TE interface as top layer	.0.15
TE interface stacked upon MPLS layer	.15.10
MPLS layer stacked upon ATM-AAL5	.10.5
ATM-AAL5 layer stacked upon ATM subinterface	.5.55
ATM subinterface stacked upon ATM	.55.72

Layer Relationship (in Descending Order)	ifStackHigherLayer/ifStackLowerLayer
ATM as bottom layer	.72.0

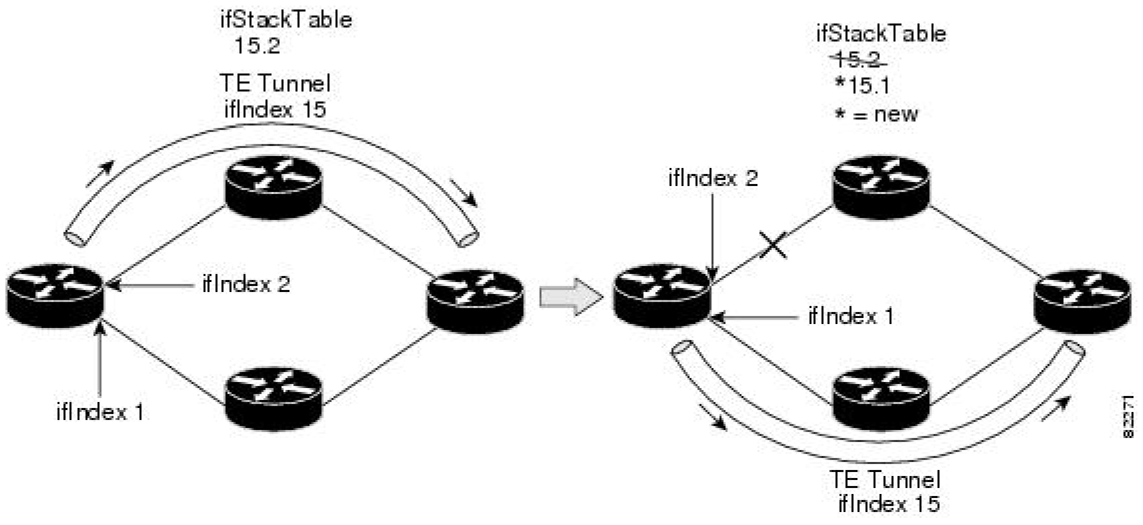
Stacking Relationships for Traffic Engineering Tunnels

MPLS TE tunnels are represented in Cisco software and the IF MIB as virtual interfaces. When properly signaled, TE tunnels pass traffic through MPLS over a physical interface. This process dictates that a TE tunnel is to be stacked on an MPLS layer that is stacked on an underlying interface.

TE tunnels can also change paths in response to different error or network conditions. These changes are instigated by using the RSVP-TE signaling protocol. When a change occurs, a tunnel can switch to a different MPLS interface. If no signaling path exists, no paths will be chosen and thus no MPLS interface will be used.

Because a TE tunnel is represented as an IF MIB ifTable entry, the ifStackTable also contains an entry corresponding to the TE tunnel. If the TE tunnel is successfully signaled, the ifStackTable also contains a link between the tunnel interface and one MPLS interface. Note that because it is possible for a TE tunnel to not have a corresponding signaled path, it is thus possible for a TE tunnel's ifStackTable entry to not have a corresponding lower layer. In this case, the lower layer variable contains the value of 0.

The figure below shows a TE tunnel before (left) and after (right) being rerouted and the effect on the ifStackTable. When ifIndex 2 fails, the TE tunnel is rerouted through ifIndex1, the 15.2 entry is removed from the ifStackTable, and the 15.1 entry is added.



MPLS Label Switching Router MIB Enhancements

All of the ifIndex references in the MPLS-LSR-MIB tables have changed from the ifIndex of the underlying physical or virtual interface to the ifIndex of the MPLS layer.

The table below shows the specific changes.

Table 42: MPLS-LSR-MIB ifIndex Objects Enhanced

Table	ifIndex
MPLS interface configuration table (mplsInterfaceConfTable)	mplsInterfaceConfIndex
MPLS in-segment table (mplsInSegmentTable)	mplsInSegmentIfIndex
MPLS cross-connect table (mplsXCTable)	mplsInSegmentIfIndex
MPLS out-segment table (mplsOutSegmentTable)	mplsOutSegmentIfIndex

The following objects from the mplsInterfaceConfTable are affected:

- mplsInterfaceOutPackets--Count only MPLS-encapsulated out packets
- mplsInterfaceInPackets--Count only MPLS-encapsulated in packets

Benefits of the MPLS Enhancements to Interfaces MIB

Improved Accounting Capability

By viewing the MPLS layer, you get MPLS-encapsulated traffic counters that do not include non-MPLS encapsulated traffic (for example, IP packets). Therefore, the counters are more useful for MPLS-related statistics.

TE Tunnel Interfaces

For TE tunnel interfaces, the stacking relationship reflects the current underlying MPLS interface that is in use and dynamically changes as TE tunnels reoptimize and reroute.

MPLS-Specific Information

The MPLS layer shows MPLS-specific information including the following:

- If MPLS is enabled
- MPLS counters
- MPLS MTU
- MPLS operational status

How to Configure MPLS Enhancements to Interfaces MIB

Enabling the SNMP Agent

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **configure terminal**
4. **snmp-server community** *string* [**view** *view-name*] [**ro** *number*]
5. **end**
6. **write memory**
7. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration of the router so that you can determine if an SNMP agent is already running on the device. 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 desired.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server community** *string* [**view** *view-name*] [**ro** *number*] **Example:** `Router(config)# snmp-server community public ro`	Configures read-only (ro) community strings for the MPLS Label Distribution Protocol (LDP) MIB. • The *string* argument functions like a password, permitting access to SNMP functionality on label switch routers (LSRs) in an MPLS network. • The optional **ro** keyword configures read-only (ro) access to the objects in the MPLS LDP MIB.

	Command or Action	Purpose

Step 5	**end**
	
	Example:
	
	`Router(config)# end`
		Exits to privileged EXEC mode.

Step 6	**write memory** **Example:** `Router# write memory`	Writes the modified SNMP configuration into NVRAM of the router, permanently saving the SNMP settings.
Step 7	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration of the router so that you can determine if an SNMP agent is already running on the device. If you see any snmp-server statements, SNMP has been enabled on the router. If any SNMP information is displayed, you can modify the information or change it as desired.

Configuration Examples for the MPLS Enhancements to Interfaces MIB

MPLS Enhancements to Interfaces MIB: Examples

The following example shows how to enable an SNMP agent:

```
Router# configure terminal
Router(config)# snmp-server community
```

In the following example, SNMPv1 and SNMPv2C are enabled. The configuration permits any SNMP manager to access all objects with read-only permissions using the community string *public*.

```
Router(config)# snmp-server community public
```

In the following example, read-only access is allowed for all objects to members of access list 4 that specify the comaccess community string. No other SNMP managers have access to any objects.

```
Router(config)# snmp-server community comaccess ro 4
```

Additional References

Related Documents

Related Topic	Document Title
SNMP commands	*Cisco IOS Network Management Command Reference*
SNMP configuration	"Configuring SNMP Support" in the *Network Management Configuration Guide.*
A description of SNMP agent support for the MPLS Traffic Engineering MIB (MPLS TE MIB)	MPLS Traffic Engineering (TE) MIB

Standards

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

MIBs

MIBs	MIBs Link
Interfaces Group MIB (IF 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

RFCs	Title
RFC 1156	*Management Information Base for Network Management of TCP/IP-based internets*
RFC 1157	*A Simple Network Management Protocol (SNMP)*
RFC 1213	*Management Information Base for Network Management of TCP/IP-based internets: MIB-II*
RFC 1229	*Extensions to the Generic-Interface MIB*

RFCs	Title
RFC 2233	*Interfaces MIB*

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 Enhancements to Interfaces MIB

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 43: Feature Information for MPLS Enhancements to Interfaces MIB

Feature Name	Releases	Feature Information
MPLS Enhancements to Interfaces MIB	12.0(23)S 12.3(8)T 12.2(33)SRA 12.2(33)SXH 12.2(33)SB Cisco IOS XE Release 2.1	This document describes the Multiprotocol Label Switching (MPLS) enhancements to the existing Interfaces MIB (RFC 2233) to support an MPLS layer. This layer provides counters and statistics specifically for MPLS. In Cisco IOS Release 12.0(23)S, this feature was introduced. This feature was integrated into Cisco IOS Release 12.3(8)T. This feature was integrated into Cisco IOS Release 12.2(33)SRA. This feature was integrated into Cisco IOS Release 12.2(33)SXH. This feature was integrated into Cisco IOS Release 12.2(33)SB. In Cisco IOS XE Release 2.1, this feature was implemented on the Cisco ASR 1000 Series Aggregation Services Routers. The following command was introduced or modified: **snmp-server community**.

Glossary

ATM -- Asynchronous Transfer Mode. The international standard for cell relay in which multiple service types (such as voice, video, or data) are conveyed in fixed-length (53-byte) cells. Fixed-length cells allow cell processing to occur in hardware, thereby reducing transit delays. ATM is designed to take advantage of high-speed transmission media, such as E3, SONET, and T3.

ATM-AAL5 --ATM adaptation layer 5. One of four AALs recommended by the ITU-T. AAL5 supports connection-oriented variable bit rate (VBR) services and is used predominantly for the transfer of classical IP over ATM and LAN emulation (LANE) traffic. AAL5 uses simple and efficient AAL (SEAL) and is the least complex of the current AAL recommendations. It offers low bandwidth overhead and simpler processing requirements in exchange for reduced bandwidth capacity and error-recovery capability.

encapsulation -- Wrapping of data in a particular protocol header. For example, Ethernet data is wrapped in a specific Ethernet header before network transit. Also, when bridging dissimilar networks, the entire frame from one network is simply placed in the header used by the data link layer protocol of the other network.

IETF --Internet Engineering Task Force. A task force (consisting of more than 80 working groups) that is developing standards for the Internet and the IP suite of protocols.

interface --The boundary between adjacent layers of the ISO model.

label --A short, fixed-length identifier that is used to determine the forwarding of a packet.

label switching--A term used to describe the forwarding of IP (or other network layer) packets using a label swapping algorithm based on network layer routing algorithms. The forwarding of these packets uses the exact match algorithm and rewrites the label.

LSR --label switching router. A device that forwards Multiprotocol Label Switching (MPLS) packets based on the value of a fixed-length label encapsulated in each packet.

MIB --Management Information Base. A database of network management information that is used and maintained by a network management protocol such as Simple Network Management Protocol (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.

MPLS --Multiprotocol Label Switching. 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.

MPLS interface--An interface on which Multiprotocol Label Switching (MPLS) traffic is enabled.

MTU --maximum transmission unit. Maximum packet size, in bytes, that a particular interface can handle.

NMS --network management system. System responsible for managing at least part of a network. An NMS is generally a reasonably powerful and well-equipped computer, such as an engineering workstation. NMSs communicate with agents to help keep track of network statistics and resources.

OID --object identifier. Values are defined in specific MIB modules. The Event MIB allows you or an NMS to watch over specified objects and to set event triggers based on existence, threshold, and Boolean tests. An event occurs when a trigger is fired; this means that a specified test on an object returns a value of true. To create a trigger, you or a network management system (NMS) configures a trigger entry in the mteTriggerTable of the Event MIB. This trigger entry specifies the OID of the object to be watched. For each trigger entry type, corresponding tables (existence, threshold, and Boolean tables) are populated with the information required for carrying out the test. The MIB can be configured so that when triggers are activated (fired) either a Simple Network Management Protocol (SNMP) Set is performed, a notification is sent out to the interested host, or both.

SNMP --Simple Network Management Protocol. A 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.

traffic engineering tunnel--A label-switched tunnel that is used for traffic engineering. Such a tunnel is set up through means other than normal Layer 3 routing; it is used to direct traffic over a path different from the one that Layer 3 routing could cause the tunnel to take.

trap --A message sent by a Simple Network Management Protocol (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.

tunnel --A secure communication path between two peers, such as routers.

CHAPTER 5

MPLS Label Switching Router MIB

The MPLS Label Switching Router MIB (MPLS-LSR-MIB) allows you to use the Simple Network Management Protocol (SNMP) to remotely monitor a label switch router (LSR) that is using the Multiprotocol Label Switching (MPLS) technology.

Scalability enhancements provided in the Cisco IOS 12.0(28)S release reduce the size of any MIB walk and improve the usability of the MPLS-LSR-MIB.

Note In Cisco IOS Release 12.2(33)SRB and Cisco IOS Release 12.2(33)SB, this MIB has been deprecated and replaced by MPLS-LSR-STD-MIB (RFC 3813). In those two releases and in later images, the entire MIB can be referenced by the name mplsLsrMIB for purposes of the SNMP server excluded/included command. If other MIB object names need to be referenced on the router, they must be referenced by MPLS-LSR-MIB::<table_entry_name>.

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.

Information About MPLS Label Switching Router MIB

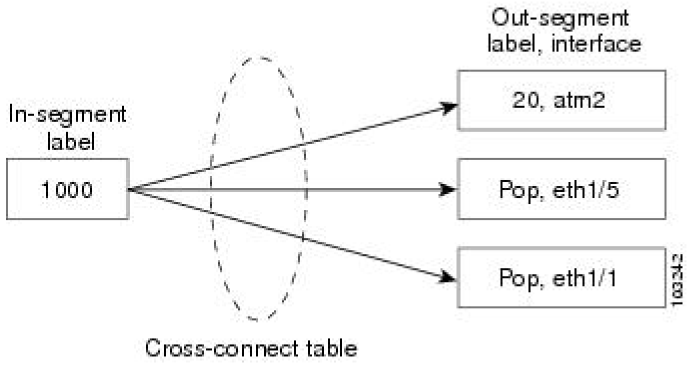
The MPLS-LSR-MIB contains managed objects that support the retrieval of label switching information from a router. The MIB is based on Revision 05 of the IETF MPLS-LSR-MIB. The MPLS-LSR-MIB mirrors a portion of the Cisco MPLS subsystem; specifically, it mirrors the Label Forwarding Information Base (LFIB). This implementation enables a network administrator to get information on the status, character, and performance of the following:

- MPLS-capable interfaces on the LSR
- Incoming MPLS segments (labels) at an LSR and their associated parameters
- Outgoing segments (labels) at an LSR and their associated parameters

In addition, the network administrator can retrieve the status of cross-connect table entries that associate MPLS segments with each other.

The figure below shows the association of the cross-connect table with incoming and outgoing segments (labels).

Figure 22: Label Forwarding with the Cross-Connect Table



Note The out-segment table does not display "no label" entries. Labels that are displayed as "POP" are the special MPLS label 3.

The notation used in the MPLS-LSR-MIB follows the conventions defined in Abstract Syntax Notation One (ASN.1). ASN.1 defines an Open System Interconnection (OSI) language used to describe data types apart from particular computer structures and presentation techniques. Each object in the MIB incorporates a DESCRIPTION field that includes an explanation of the object's meaning and usage, which, together with the other characteristics of the object (SYNTAX, MAX-ACCESS, and INDEX) provides sufficient information for management application development, as well as for documentation and testing.

The MPLS-LSR-MIB represents an ASN.1 notation reflecting an idealized MPLS LSR.

A network administrator can access the entries (objects) in the MPLS-LSR-MIB by means of any SNMP-based network management system (NMS). The network administrator can retrieve information in the MPLS-LSR-MIB using standard SNMP **get** and **getnext** operations.

Typically, SNMP runs as a low-priority process. The response time for the MPLS-LSR-MIB is expected to be similar to that for other MIBs. The size and structure of the MIB and other MIBs in the system influence response time when you retrieve information from the management database. Traffic through the LSR also affects SNMP performance. The busier the switch is with forwarding activities, the greater the possibility of lower SNMP performance.

MPLS-LSR-MIB Elements

The top-level components of the MPLS-LSR-MIB consist of

- Tables and scalars (mplsLsrObjects)
- Traps (mplsLsrNotifications and mplsLsrNotifyPrefix)
- Conformance (mplsLsrConformance)

This Cisco implementation does not support the notifications defined in the MIB, nor does it support the labelStackTable or the trafficParamTable.

MPLS-LSR-MIB Tables

The Cisco implementation of the MPLS-LSR-MIB supports four main tables:

- Interface configuration
- In-segment
- Out-segment
- Cross-connect

The MIB contains three supplementary tables to supply performance information. This implementation does not support the label stack and traffic parameter tables.

The following sections list the MPLS-LSR-MIB tables (main and supplementary), their functions, table objects that are supported, and table objects that are *not* supported.

MPLS interface configuration table (mplsInterfaceConfTable)

Provides information for each MPLS-capable interface on an LSR.

Supports:

- A unique interface index or zero
- Minimum and maximum values for an MPLS label received on the interface
- Minimum and maximum values for an MPLS label sent from the interface
- A value for an MPLS label sent from the interface
- Per platform (0) or per interface (1) setting
- The storage type

Does not support:

- The total usable bandwidth on the interface
- The difference between the total usable bandwidth and the bandwidth in use

MPLS interface performance table (mplsInterfacePerfTable)

Augments the MPLS interface configuration table.

Supports:

- The number of labels in the incoming direction in use
- The number of top-most labels in outgoing label stacks in use

Does not support:

- The number of top-most labels in outgoing label stacks in use
- The number of labeled packets discarded because no cross-connect entries exist
- The number of outgoing MPLS packets requiring fragmentation for transmission

MPLS in-segment table (mplsInSegmentTable)

Contains a description of incoming segments (labels) at an LSR and their associated parameters.

Administrative and operational status objects for this table control packet transmission. If administrative and operational status objects are down, the LSR does not forward packets. If these status objects are up, the LSR forwards packets.

Supports:

- A unique index identifier
- The incoming label
- The number of labels to pop from the incoming segment
- An address family number from the Internet Assigned Number Authority (IANA)
- A segment cross-connect entry association
- The segment owner
- The storage type
- The administrative status
- The operational status

Note The administrative status and operational status are always up for inSegments in the Cisco implementation. Otherwise, these entries do not appear in the table.

Does not support:

- A pointer to a traffic parameter table entry (set to the default 0.0)

MPLS in-segment performance table (mplsInSegmentPerfTable)

Augments the MPLS in-segment table, providing performance information and counters for incoming segments on an LSR.

Supports:

- The number of 32-bit octets received
- The number of 64-bit octets received
- The time of the last system failure that corresponded to one or more incoming segment discontinuities

Note The lastFailure parameter is set to zero because it has no meaning in the Cisco implementation.

Does not support:

- The total number of packets received
- The number of packets with errors
- The number of labeled packets discarded with no errors

MPLS out-segment table (mplsOutSegmentTable)

Contains a description of outgoing segments from an LSR and their associated parameters.

Administrative and operational status objects for this table control packet transmission. If administrative and operational status objects are down, the LSR does not forward packets. If these values are up, the LSR forwards packets.

Supports:

- A unique index identifier
- An interface index of the outgoing interface
- An indication of whether or not a top label is pushed onto the outgoing packet's label stack
- The label to push onto the outgoing packet's label stack (if the previous value is true)
- The next hop address type
- The IPv4 address of the next hop
- The segment cross-connect entry association
- The segment owner
- The storage type
- The administrative status
- The operational status

Note The administrative and operational status entries are always up in the Cisco implementation. Otherwise, the administrative and operational status entries do not appear in the table.

Does not support:

- An IPv6 address of the next hop
- A pointer to a traffic parameter table entry (set to the default 0.0)

MPLS out-segment performance table (mplsOutSegmentPerfTable)

Augments the MPLS out-segment table, providing performance information and counters for outgoing segments on an LSR.

Supports:

- The number of 32-bit octets sent
- The number of 64-bit octets sent
- The time of the last system failure that corresponded to one or more outgoing segment discontinuities

Does not support:

- The number of packets sent
- The number of packets that could not be sent because of errors
- The number of packets discarded with no errors

MPLS cross-connect table (mplsXCTable)

Associates inSegments (labels) with outSegments (labels) to show the manager how the LSR is currently swapping these labels.

A row in this table consists of one cross-connect entry that is indexed by the cross-connect index, the interface index of the incoming segment, the incoming label, and the out-segment index.

The administrative and operational objects for this table control packet forwarding to and from a cross-connect entry (XCEntry). The administrative status and operational status are always up in the Cisco implementation. Otherwise, the LSR would not forward packets.

Supports:

- A unique index identifier for a group of cross-connect segments
- A label switched path (LSP) to which the cross-connect entry belongs
- An index to the MPLS label stack table that identifies the stack of labels to be pushed under the top label
- An indication whether or not to restore the cross-connect entry after a failure (the default value is false)
- The cross-connect owner
- The storage type
- The administrative status (if up)
- The operational status (if up)

Note The administrative status and operational status are always up in the Cisco implementation. Otherwise, these status entries do not appear in the table.

Does not support:

- Tunnel IDs as label switched path (LSP) ID objects

Information from Scalar Objects

The MPLS-LSR-MIB supports several scalar objects. In the Cisco implementation of the MIB, the following scalar objects are hard-coded to the value indicated and are read-only objects:

- mplsOutSegmentIndexNext (0)--The value for the out-segment index when an LSR creates a new entry in the MPLS out-segment table. The 0 indicates that this is not implemented because modifications to this table are not allowed.
- mplsXCTIndexNext (0)--The value for the cross-connect index when an LSR creates an entry in the MPLS cross-connect table. The 0 indicates that no unassigned values are available.
- mplsMaxLabelDepth(2)--The value for the maximum stack depth.
- mplsLabelStackIndexNext (0)--The value for the label stack index when an LSR creates entries in the MPLS label stack table. The 0 indicates that no unassigned values are available.
- mplsTrafficParamIndexNext (0)--The value for the traffic parameter index when an LSR creates entries in the MPLS traffic parameter table. The 0 indicates that no unassigned values are available.

The following scalar objects do not contain information for the MPLS-LSR-MIB and are coded as false:

- mplsInSegmentTrapEnable (false)--In-segment traps are not sent when this value is false.
- mplsOutSegmentTrapEnable (false)--Out-segment traps are not sent when this value is false.
- mplsXCTrapEnable (false)--Cross-connect traps are not sent when this value is false.

No trap information exists to support the MIB. Therefore, the following traps are not supported:

- mplsInSegmentUp
- mplsInSegmentDown
- mplsOutSegmentUp
- mplsOutSegmentDown
- mplsXCUp
- mplsXCDown

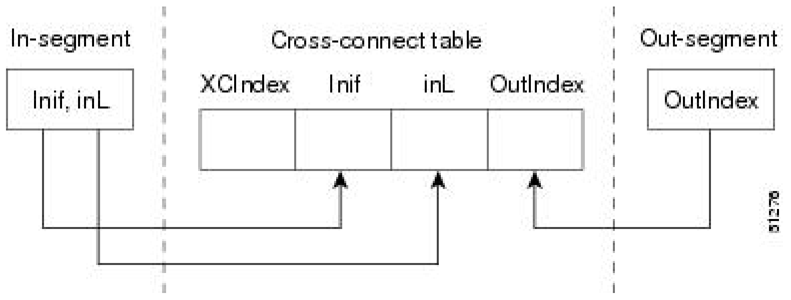
Linking Table Elements

In the cross-connect table, cross-connect entries associate incoming segments and interfaces with outgoing segments and interfaces. The following objects index the cross-connect entry:

- Cross-connect index--A unique identifier for a group of cross-connect entries in the cross-connect table. In the Cisco implementation, this value is always the same as that for the outSegmentIndex, unless there is no label or if the label has been popped.
- Interface index of the in-segment--A unique index for an entry in the in-segment table that represents an incoming MPLS interface. The value 0 means platform wide, for any entries that apply to all interfaces.
- Incoming label--An entry in the in-segment table that represents the label on the incoming packet.
- Out-segment index--A unique identifier for an entry in the out-segment table that contains a top label for the outgoing packet's label stack and an interface index of the outgoing interface.

The figure below shows the links between the in-segment and the out-segment in the cross-connect table.

Figure 23: Cross-Connect Table Links



The table below shows the cross-connect table links you might see in the output from SNMP **get** operations on the MPLS-LSR-MIB objects that index a cross-connect entry. These objects include

- In-Segment Values--mplsInSegmentIfIndex and mplsInSegmentLabel
- Cross-Connect Entry--mplsXCIndex
- Out-Segment Values--mplsOutSegmentIndex

Table 44: MPLS LSR Output Showing Cross-Connect Table Links

In-Segment Values	Cross-Connect Entry	Out-Segment Values
0[1], 1000	500[2], 0, 1000, 0Linking Table Elements, on page 177	--
	501, 0, 1000, 501	501 = Pop (topLabel), Eth 1/5
	502, 0, 1000, 502	502 = Pop (topLabel), Eth, 1/1

1 All MPLS-enabled interfaces can receive incoming labels.

2 For this implementation of the MPLS-LSR-MIB, the cross-connect index and the out-segment index are the same. If there is no outsegment, the value will be zero.

Note The OutSegmentIndex object is not the label. The label can be retrieved from the mplsOutSegmentTopLabel object.

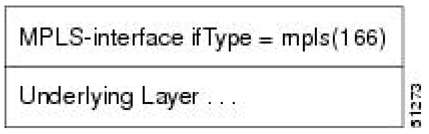
Interface Configuration Table and Interface MIB Links

The MPLS interface configuration table lists interfaces that support MPLS technology. An LSR creates an entry dynamically in this table for each MPLS-capable interface. An interface becomes MPLS-capable when MPLS is enabled on that interface. A non-zero index for an entry in this table points to the ifIndex for the corresponding interface entry in the MPLS-layer in the ifTable of the Interfaces Group MIB.

The ifTable contains information on each interface in the network. Its definition of an interface includes any sublayers of the internetwork layer of the interface. MPLS interfaces fit into this definition of an interface. Therefore, each MPLS-enabled interface is represented by an entry in the ifTable.

The interrelation of entries in the ifTable is defined by the interfaces stack group of the Interfaces Group MIB. The figure below shows how the stack table might appear for MPLS interfaces. The underlying layer refers to any interface that is defined for MPLS internetworking, for example, ATM, Frame Relay, or Ethernet.

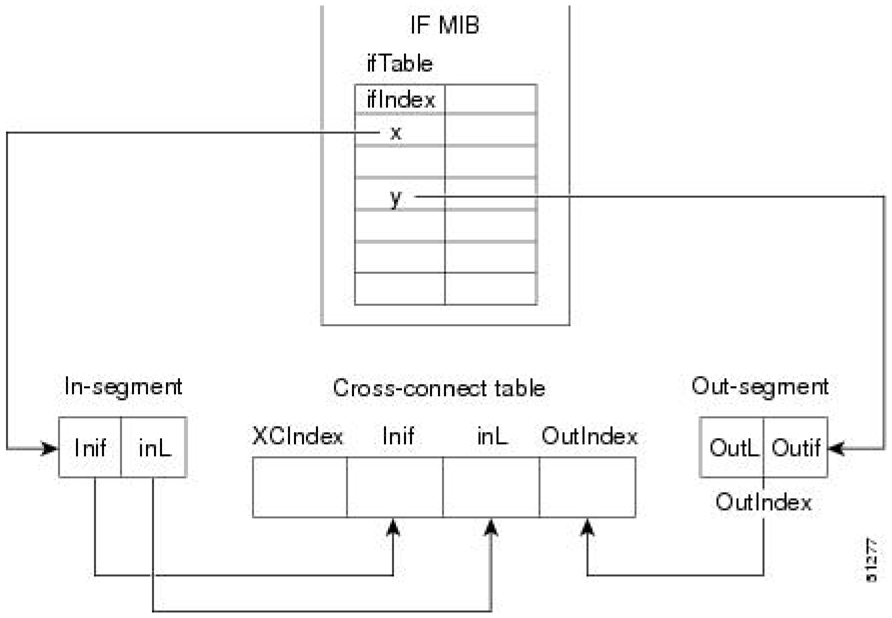
Figure 24: Interface Group MIB Stack Table for MPLS Interfaces



Note Tunnel interfaces are included in the MPLS list for the current implementation.

The incoming and outgoing packets include a reference to the interface index for the ifTable of the Interfaces Group MIB. The figure below shows the links between MPLS-LSR-MIB objects and the Interfaces Group MIB.

Figure 25: MPLS-LSR-MIB and Interfaces Group MIB Links



- For the Interfaces Group MIB (IF MIB):
  - ifTable represents the MPLS interface table.
  - ifIndex represents the index to an entry in the MPLS interface table.
- For the In-segment:
  - Inif represents the interface on the incoming segment (references an index entry in the ifTable).
  - inL represents the label on the incoming segment.
- For the Out-segment:
  - OutL represents the label on the outgoing segment.
  - Outif represents the interface on the outgoing segment (references an index entry in the ifTable).
- For the Cross-connect table:
  - XCIndex represents the index to an entry in the MPLS cross-connect table.
  - Inif represents the interface on the incoming segment.
  - inL represents the MPLS label on the incoming segment.
  - OutIndex represents an index to an entry in the MPLS out-segment table.

Using the MPLS-LSR-MIB

The MPLS-LSR-MIB enables you to display the contents of the MPLS Label Forwarding Information Base (LFIB). It gives you the same information that you can obtain using the CLI command **show mpls forwarding-table**.

However, the MPLS-LSR-MIB approach offers these advantages over the CLI command approach:

- A more efficient use of network bandwidth
- Greater interoperability among vendors
- Greater security (SMNP Version 3)

The following paragraphs describe the MPLS-LSR-MIB structure and show, through the use of an example, how the two approaches to the information display compare.

MPLS-LSR-MIB Structure

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.

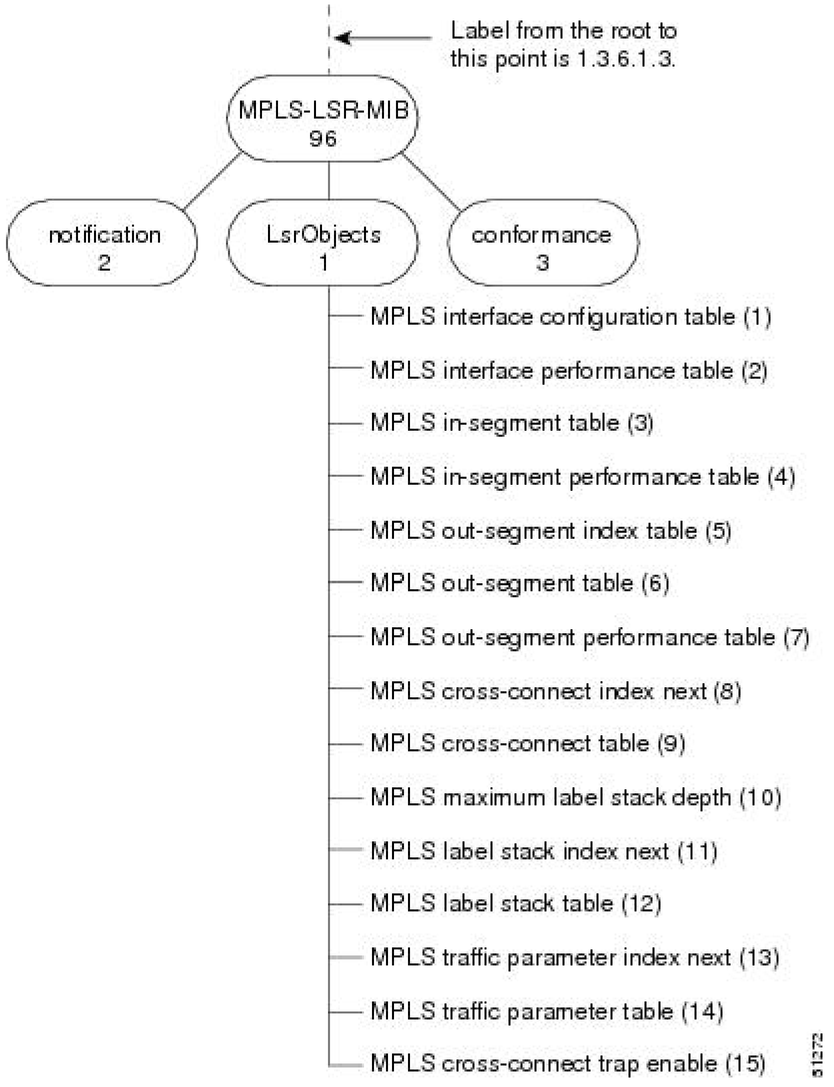
The MPLS-LSR-MIB falls on the experimental branch of the Internet MIB hierarchy. The experimental branch of the Internet MIB hierarchy is represented by the object identifier 1.3.6.1.3. This branch can also be represented by its object name *iso.org.dod.internet.experimental* . The MPLS-LSR-MIB is identified by the object name *mplsLsrMIB* , which is denoted by the number 96. Therefore, objects in the MPLS-LSR-MIB can be identified in either of the following ways:

- The object identifier--1.3.6.1.3.96.[MIB-variable]
- The object name--*iso.org.dod.internet.experimental.mplsLsrMIB.[MIB-variable]*

To display a *MIB-variable* , you enter an SNMP **get** command with an object identifier. Object identifiers are defined by the MPLS-LSR-MIB.

The figure below shows the position of the MPLS-LSR-MIB in the Internet MIB hierarchy.

Figure 26: MPLS-LSR-MIB in the Internet MIB Hierarchy


Label from the root to this point is 1.3.6.1.3.
MPLS-LSR-MIB 96
notification 2
LsrObjects 1
conformance 3
MPLS interface configuration table (1)
MPLS interface performance table (2)
MPLS in-segment table (3)
MPLS in-segment performance table (4)
MPLS out-segment index table (5)
MPLS out-segment table (6)
MPLS out-segment performance table (7)
MPLS cross-connect index next (8)
MPLS cross-connect table (9)
MPLS maximum label stack depth (10)
MPLS label stack index next (11)
MPLS label stack table (12)
MPLS traffic parameter index next (13)
MPLS traffic parameter table (14)
MPLS cross-connect trap enable (15)
51272

CLI Commands and the MPLS-LSR-MIB

The MPLS LFIB is the component of the Cisco MPLS subsystem that contains management information for LSRs. You can access this management information by means of either of the following:

- Using the **show mpls forwarding-table** CLI command
- Entering SNMP **get** commands on a network manager

The following examples show how you can gather LSR management information using both methods.

CLI Command Output

A **show mpls forwarding-table** CLI command allows you to look at label forwarding information for a packet on a specific MPLS LSR.

```
Router# show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Tag     Outgoing   Next Hop
Tag    Tag or VC     or Tunnel Id      Switched      interface
19     Pop Tag       10.3.4.0/24       0             Et1/4      10.22.23.23
22     23            14.14.14.14/32    0             AT2/0.1    point2point
       1/36          14.14.14.14/32    0             AT2/0.2    point2point
```

MPLS-LSR-MIB Output

SNMP commands on MIB objects also allow you to look at the label forwarding information for a specific MPLS LSR.

You can do a walk-through of the MIB by running a command such as **getmany -v2c public mplsLsrMIB** on a network manager where **getmany** does repeated SNMP **getnext** operations to retrieve the contents of the MPLS-LSR-MIB.

```
mplsXCOperStatus.9729.0.19.9729 = up(1)
mplsXCOperStatus.11265.0.22.11265 = up(1)
mplsXCOperStatus.11266.0.22.11266 = up(1)
```

You can continue to scan the output of the **getmany** command for the following (from the MPLS out-segment table):

- Out-segment's top label objects (mplsOutSegmentTopLabel)

```
mplsOutSegmentTopLabel.9729 = 3
mplsOutSegmentTopLabel.11265 = 23
mplsOutSegmentTopLabel.11266 = 65572
```

Note 65572 is 1/36 in label form (1 is the high-order 16 bits. 36 is the low-order 16 bits.)

- Out-segment's interface index (mplsOutSegmentIfIndex)

```
mplsOutSegmentIfIndex.9729 = 7
mplsOutSegmentIfIndex.11265 = 28
mplsOutSegmentIfIndex.11266 = 31
```

Benefits

The benefits described in the following paragraphs are available to you with the MPLS-LSR-MIB.

Troubleshooting LSR Problems

By monitoring the cross-connect entries and the associated incoming and outgoing segments, you can see which labels are installed and how they are being swapped. Use the MPLS-LSR-MIB in place of the **show mpls forwarding** CLI command.

Monitoring of LSR Traffic Loads

By monitoring interface and packet operations on an MPLS LSR, you can identify high- and low-traffic patterns, as well as traffic distributions.

Improvement of Network Performance

By identifying potentially high-traffic areas, you can set up load sharing to improve network performance.

Verification of LSR Configuration

By comparing results from SNMP **get** commands and the **show mpls forwarding** CLI command, you can verify your LSR configuration.

Displaying of Active Label Switched Paths

By monitoring the cross-connect entries and the associated incoming segments and outgoing segments, you can determine the active LSPs.

How to Configure the MPLS LSR MIB

Prerequisites

The MPLS-LSR-MIB requires the following:

- SNMP installed and enabled on the LSR
- MPLS enabled on the LSR
- 60K of memory

Note Additional capacity is not required for runtime dynamic random-access memory (DRAM).

Enabling the SNMP Agent

The SNMP agent for the MPLS-LSR-MIB is disabled by default. To enable the SNMP agent, perform the following steps:

SUMMARY STEPS

1. **enable**
2. show running-config
3. **configure terminal**
4. **snmp-server community** *string* [**view** *view-name*] [**ro**] [*number*]
5. **end**
6. copy running-config startup-config

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	show running-config **Example:** `Router# show running-config`	Displays the running configuration of the router to determine if an SNMP agent is already running on the device. 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 desired.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server community** *string* [**view** *view-name*] [**ro**] [*number*] **Example:** `Router(config)# snmp-server community public ro`	Configures read-only (ro) SNMP community strings. This command enables the SNMP agent and permits any SNMP manager to access all objects with read-only permission using the community string public.
Step 5	**end** **Example:** `Router(config)#` **end**	Exits to privileged EXEC mode.
Step 6	copy running-config startup-config **Example:** `Router# copy running-config startup-config`	Copies the modified SNMP configuration into router NVRAM, permanently saving the SNMP settings. When you are working with Cisco IOS Release 10.3 or earlier, use the **write memory** command.

Verifying That the SNMP Agent Has Been Enabled

To verify that the SNMP agent has been enabled, perform the following steps:

SUMMARY STEPS

1. Access the router through a Telnet session:
2. Enter privileged mode:
3. Display the running configuration and look for SNMP information:

DETAILED STEPS

Step 1 Access the router through a Telnet session:

Example:

```
Prompt# telnet xxx.xxx.xxx.xxx
```

where *xxx.xxx.xxx.xxx* represents the IP address of the target device.

Step 2 Enter privileged mode:

Example:

```
Router# enable
```

Step 3 Display the running configuration and look for SNMP information:

Example:

```
Router# show running-configuration
...
...
snmp-server community public RO
```

If you see any “snmp-server” statements, SNMP has been enabled on the router.

Configuration Examples for the MPLS LSR MIB

The following example shows how to enable an SNMP agent.

```
configure terminal
snmp-server community
```

In the following example, SNMPv1 and SNMPv2C are enabled. The configuration permits any SNMP manager to access all objects with read-only permissions using the community string *public* .

```
configure terminal
snmp-server community public
```

In the following example, read-only access is allowed for all objects to members of access list 4 that specify the *comaccess* community string. No other SNMP managers have access to any objects.

```
configure terminal
nmp-server community comaccess ro 4
```

Additional References

Related Documents

Related Topic	Document Title
Configuring SNMP using Cisco IOS software	• *Network Management Configuration Guide* . Configuring SNMP Support • Network Management Command Reference, SNMP Commands

Standards

Standard	Title
draft-ietf-mpls-lsr-mib-05.txt	MPLS Label Switch Router Management Information Base Using SMIv2
draft-ietf-mpls-arch-07.txt	Multiprocol Label Switching Architecture

MIBs

MIBs	MIBs Link
• MPLS Label Switching Router MIB (MPLS-LSR-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

RFCs	Title
The LSR implementation supporting the MPLS-LSR-MIB is in full compliance with all provisions of Section 10 of RFC 2026.	*The Internet Standards Process*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content. 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 Label Switching Router MIB

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 45: Feature Information for MPLS Label Switching Router MIB

Feature Name	Releases	Feature Information
MPLS Label Switching Router MIB	12.0(14)ST 12.2(2)T 12.0(22)S 12.2(14)S 12.2(25)S 12.0(28)S 12.2(33)SRB 12.2(33)SB	The MPLS Label Switching Router MIB (MPLS-LSR-MIB) allows you to use the Simple Network Management Protocol (SNMP) to remotely monitor a label switch router (LSR) that is using the Multiprotocol Label Switching (MPLS) technology. This feature was introduced on Cisco IOS Release 12.0(14)ST. This feature was integrated into Cisco IOS Release 12.2(2)T. This feature was implemented on the Cisco 12000 series routers and integrated into Cisco IOS Release 12.0(22)S. This feature was integrated into Cisco IOS Release 12.2(14)S and implemented on Cisco 7200 and Cisco 7500 series routers. This feature was updated to include scalability enhancements in Cisco IOS Release 12.0(28)S. In Cisco IOS Release 12.2(25)S, this feature was updated to work in the MPLS High Availability environment with the Cisco 7500 series routers. In Cisco IOS Release 12.2(33)SRB, this MIB has been deprecated and replaced by MPLS-LSR-STD-MIB (RFC 3813). In Cisco IOS Release 12.2(33)SB, this MIB has been deprecated and replaced by MPLS-LSR-STD-MIB (RFC 3813). No commands were introduced or modified.

Glossary

cross-connect (XC) --An association of in-segments and incoming Multiprotocol Label Switching (MPLS) interfaces to out-segments and outgoing MPLS interfaces.

IETF --Internet Engineering Task Force. A task force (consisting of more that 80 working groups) that is developing standards for the Internet and the IP suite of protocols.

inSegment --A label on an incoming packet that is used to determine the forwarding of the packet.

Internet Engineering Task Force --See IETF.

label --A short, fixed length identifier that is used to determine the forwarding of a packet.

Label Distribution Protocol --See LDP.

label switched path --See LSP.

label switching --Describes the forwarding of IP (or other network layer) packets by a label swapping algorithm based on network layer routing algorithms. The forwarding of these packets uses the exact match algorithm and rewrites the label.

label switch router --See LSR.

LDP --Label Distribution Protocol. A standard protocol that operates between Multiprotocol Label Switching (MPLS)-enabled routers to negotiate the labels (addresses) used to forward packets. The Cisco proprietary version of this protocol is the Tag Distribution Protocol (TDP).

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 Multiprotocol Label Switching (MPLS) packets based on the value of a fixed-length label encapsulated in each packet.

Management Information Base --See MIB.

MIB --Management Information Base. A database of network management information that is used and maintained by a network management protocol such as Simple Network Management Protocol (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.

MPLS --Multiprotocol Label Switching. A switching method that forwards IP traffic through use of a label. This label instructs the routers and the switches in the network where to forward the packets. The forwarding of MPLS packets is based on preestablished IP routing information.

MPLS interface --An interface on which Multiprotocol Label Switching (MPLS) traffic is enabled.

Multiprotocol Label Switching --See MPLS.

notification request --A message sent by a Simple Network Management Protocol (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.

outSegmen t--A label on an outgoing packet.

Simple Network Management Protocol --See SNMP.

SNMP --Simple Network Management Protocol. A 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 --A message sent by a Simple Network Management Protocol (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.

Note Refer to the Cisco Dictionary of Internetworking Terms and Acronyms for terms not included in this glossary.

CHAPTER 6

MPLS Label Distribution Protocol MIB

This document describes the Simple Network Management Protocol (SNMP) agent support provided in Cisco software for the MPLS Label Distribution Protocol Management Information Base (MPLS LDP MIB).

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.

Restrictions for MPLS LDP MIB

The MPLS LDP MIB is limited to read-only (RO) permission for MIB objects, except for MIB object mplsLdpSessionUpDownTrapEnable, which is writable by the SNMP agent.

Setting this object to a value of true enables both the mplsLdpSessionUp and mplsLdpSessionDown notifications on the Label Switched Router (LSR); conversely, setting this object to a value of false disables both of these notifications. The value of the mplsLdpSessionUpDownTrapEnable object is stored in NVRAM on the MPLS LDP MIB host.

For a description of notification events, see the Events Generating MPLS LDP MIB Notifications section.

Most MPLS LDP MIB objects are set up automatically during the LDP peer discovery (Hello) process and the subsequent negotiation of parameters and establishment of LDP sessions between the LDP peers.

Information About MPLS LDP MIB

MPLS LDP Overview

Multiprotocol Label Switching (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 label switching routers (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.

By means of LDP, LSRs can collect, distribute, and release 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 LDP MIB Overview

The MPLS LDP MIB has been implemented to enable standard, SNMP-based network management of the label switching features in Cisco 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 MPLS LDP MIB.

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

By means of an SNMP agent, you can access MPLS LDP MIB objects using standard SNMP **get** operations to accomplish a variety of network management tasks. All the objects in the MPLS LDP MIB follow the conventions defined in the Internet Engineering Task Force (IETF) draft MIB entitled *draft-ietf-mpls-ldp-mib-08.txt,* which defines network management objects in a structured and standardized manner. This draft MIB is continually evolving toward the status of a standard. Accordingly, the MPLS LDP MIB will be implemented in a manner that tracks the evolution of this IETF document.

Slight differences that exist between the IETF draft MIB and the implementation of equivalent functions in Cisco software require some minor translations between the MPLS LDP MIB objects and the internal data structures of Cisco software. Such translations are accomplished by the SNMP agent, which runs in the background on the NMS workstation as a low priority process.

The MPLS LDP MIB provides the following functions:

- The MPLS LDP MIB can generate and send event notification messages to signal changes in the status of LDP sessions.
- You can enable and disable event notification messages by using SNMP CLI commands.
- You can specify the name or the IP address of an NMS workstation where event notification messages are sent to serve network administrative and management purposes.
- You can store the configuration pertaining to an event notification message in nonvolatile memory (NVRAM) of the NMS.

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

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

Note Because the MPLS LDP MIB was not given an Internet Assigned Numbers Authority (IANA) Experimental OID at the time of its implementation, Cisco chose to implement the MIB under the Cisco Experimental OID number: ciscoExperiment 1.3.6.1.4.1.9.10 mplsLdpMIB 1.3.6.1.4.1.9.10.65 If the MPLS LDP MIB is assigned an IANA Experimental OID number, Cisco will deprecate all objects in the MIB under the Cisco Experimental OID and reposition the objects under the IANA Experimental OID.

Benefits of Using MPLS LDP MIB

The MPLS LDP MIB provides the following benefits:

- Establishing LDP sessions between peer devices in an MPLS network
- Retrieving 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 VPI/VCI pairs to be used in forming labels
- Gathering statistics related 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
- Monitoring the time remaining for Hello adjacencies
- Monitoring the characteristics and status of LDP peers, such as:

  - Type of internetwork layer address of LDP peers
  - Actual internetwork layer address of LDP peers
  - Default MTU of the LDP peer
  - Number of seconds the LDP peer proposes as the value of the KeepAlive interval
  - Establishment of VPI/VCI label ranges to be made known to LDP peers

- Monitoring the characteristics and status of LDP sessions, such as:
  - 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 or not)
  - Determining the range of VPI/VCI pairs to be used by an LDP session
  - Determining the last active interface of an LDP session

Description of MPLS LDP MIB Elements

The MPLS LDP MIB includes the following elements:

- LDP entity--Relates to an instance of LDP for purposes of exchanging label spaces.
- 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 which affirms the state of two LSRs in an MPLS network as being adjacent to each other (that is, as being LDP peers).

A Hello adjacency constitutes the working context between two LSRs in an MPLS network. The adjacency is used for the exchange of label binding information.

These MPLS LDP MIB elements are briefly described under separate headings below.

In effect, the MPLS LDP MIB provides a network management database that supports real-time access to the various MIB objects within, reflecting 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 MIB supports 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 having the object name *mplsLdpEntityLdpId* . This object consists of the router ID (four octets) and an interface number (two octets). The router ID encodes an IP address assigned to the LSR. The interface number identifies a specific label space available within the LSR.

An LDP entity represents a label space that is targeted for distribution to an LDP peer. In the case of an interface-specific LDP entity, the label space is distributed to a single LDP peer by means of a single LDP session.

Conversely, a platform-wide LDP entity can be associated with multiple LDP peers. In this case, the label space is distributed to multiple LDP peers by means of a separate LDP session pertaining to each peer.

LDP Peers

If an LSR has a label space to advertise to another LSR, or to multiple LSRs, there would be one LDP session for each LSR receiving the label space information. The receiver of the label space information is referred to as an LDP peer.

Per-interface label spaces are advertised to a single LDP peer by means of a single LDP session. *Per-platform* label spaces are advertised to multiple LDP peers by means of multiple LDP sessions.

The possible existence of multiple per-platform LDP peers dictates not only that an LDP entity be identified by its unique LDP tag, but also by its LDP index. In this case, the label space is the same, but the LDP Index differentiates the LDP session over which the label space is distributed to multiple LDP peers.

LDP Sessions

LDP sessions between local entities and remote peers distribute label spaces. There is always a one-to-one correspondence between an LDP peer and an LDP session. A single LDP session is a label distribution protocol instance that communicates across one or more network links with a single LDP peer. In the case of a platform-wide local LDP entity, there may be multiple LDP sessions and a corresponding number of remote LDP peers.

LDP Hello Adjacencies

An LDP session is an LDP instance that communicates across one or more network links to a peer protocol instance. An LDP Hello adjacency exists for each link on which LDP runs. Multiple link adjacencies exist whenever there are multiple links to the same LDP peer. In the case of a platform-wide label space, for example, there is a separate LDP peer/LDP session relationship for each LSR to which a label space may be advertised.

MPLS LDP MIB Object Categories

The MPLS LDP MIB contains numerous definitions of managed objects for the MPLS Label Distribution Protocol, as defined in the IETF draft document entitled *draft-ietf-mpls-ldp-08.txt* .

The managed objects in the MPLS LDP MIB are structured according to the following categories:

- MPLS LDP Textual Conventions

- MPLS LDP Objects
- MPLS Label Distribution Protocol Entity Objects
- LDP Entity Objects for Generic Labels
- LDP Entity Objects for ATM
- MPLS LDP Entity Configured ATM Label Range Table
- MPLS Entity Objects for Frame Relay
- Frame Relay Label Range Components
- MPLS LDP Entity Statistics Table
- MPLS LDP Entity Peer Table
- MPLS LDP Hello Adjacency Table
- MPLS LDP Sessions Table
- MPLS LDP ATM Session Information
- MPLS LDP Frame Relay Session Information
- MPLS LDP Session Statistics Table
- Address Message/Address Withdraw Message Information
- MPLS LDP LIB Table
- MPLS LDP FEC Table
- Notifications
- Module Conformance Statement

Events Generating MPLS LDP MIB Notifications

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

The MPLS LDP MIB objects that announce LDP status transitions and event notifications include 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).
- mplsLdpSessionDown--This message is generated when an LDP session between a local LSR and its adjacent LDP peer is terminated.

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.

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.

- mplsLdpFailedInitSessionThresholdExceeded--This message is generated when a local LSR and an adjacent LDP peer attempt to set up an LDP session between them, but fail to do so after a specified number of attempts. The default number of attempts is 8. This default value is implemented in Cisco software and cannot be changed by either the CLI or an SNMP agent.

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/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.

Operationally, 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.

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

- • Nonoverlapping ATM VPI/VCI ranges (as noted above) or nonoverlapping Frame-Relay DLCI ranges between LSRs attempting to set up an LDP session
  - Unsupported label distribution method
  - Dissimilar protocol data unit (PDU) sizes
  - Dissimilar LDP feature support

How to Configure MPLS LDP MIB

Enabling the SNMP Agent for the MPLS LDP MIB

By default, the SNMP agent for the MPLS LDP MIB is disabled. To enable the SNMP agent on the host NMS workstation, perform the following procedure.

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** [**interface** | **map-class**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • 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`	Sets up 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 and 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 only able to 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.

	Command or Action	Purpose

Step 5	**do copy running-config startup-config**
	
	Example:
	
	`Router(config)# do copy running-config startup-config`
		Saves the modified configuration to nonvolatile memory (NVRAM) as the startup configuration file.
		• The **do** command allows you to perform EXEC level commands in configuration mode.

Step 6	**exit** **Example:** `Router(config)# exit`	Returns to privileged EXEC mode.
Step 7	**show running-config** [**interface** | **map-class**] **Example:** `Router# show running-config	include smnp-server`	(Optional) Displays the configuration information currently on the router, the configuration for a specific interface, or map-class information. • Use the **show running-config** command to check that the **snmp-server** statements appear in the output.

Configuring the Router to Send SNMP Traps

Perform this task to configure the router to sendm SNMP traps to a host.

The **snmp-server host** command specifies which hosts receive 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.

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-addr* [**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 vpn** [**illegal-label**] [**max-thresh-cleared**] [**max-threshold**] [**mid-threshold**] [**vrf-down**] [**vrf-up**]
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-addr* [**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-vpn`	Specifies the recipient of an SNMP notification operation. • The*host-addr* 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 and argument pair names the User Datagram Protocol (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. • The **vrf** *vrf-name* keyword and argument pair specifies the VRF table that should be used to send SNMP notifications.
Step 4	**snmp-server enable traps mpls vpn** [**illegal-label**] [**max-thresh-cleared**] [**max-threshold**] [**mid-threshold**] [**vrf-down**] [**vrf-up**]	Enables the router to send MPLS VPN-specific SNMP notifications (traps and informs). • The **illegal-label** keyword enables a notification for any illegal labels received on a VRF interface. Labels are illegal if they are outside the legal range, do not have an LFIB entry, or do not match table IDs for the label.

	Command or Action	Purpose
	Example: `Router(config)# snmp-server enable traps mpls vpn vrf-down vrf-up`	• The **max-thresh-cleared** keyword enables a notification when the number of routes falls below the limit after the maximum route limit was attempted. • The **max-threshold** keyword enables a notification that a route creation attempt was unsuccessful because the maximum route limit was reached. Another MplsNumVrfRouteMaxThreshExceeded notification is not sent until the number of routes falls below the maximum threshold and reaches the maximum threshold again. The max-threshold value is determined by the **maximum routes** command in VRF configuration mode. • The **mid-threshold** keyword enables a notification of a warning that the number of routes created has crossed the warning threshold. This warning is sent only at the time the warning threshold is exceeded. • The **vrf-down** keyword enables a notification for the removal of a VRF from an interface or the transition of an interface to the down state. • The **vrf-up** keyword enables a notification for the assignment VRF to an interface that is operational or for the transition of a VRF interface to the operationally up state.

Step 5	**end**
	Example:
	`Router(config)# end`
		(Optional) Exits to privileged EXEC mode.

Verifying the Status of the SNMP Agent

To verify that the SNMP agent has been enabled on the host NMS workstation, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **exit**

DETAILED STEPS

Step 1 **enable**
Use this command to enable privileged EXEC mode. Enter your password if prompted. For example:

Example:

```
Router> enable
Router#
```

Step 2 **show running-config**

Use this command to display the running configuration on the host NMS and examine the output for SNMP information. For example:

Example:

```
Router# show running-config
.
.
.
snmp-server community public RO
snmp-server community private RO
```

The presence of any snmp-server statement in the output that takes the form shown above verifies that the SNMP agent has been enabled on the host NMS workstation.

Step 3 **exit**

Use this command to exit to user EXEC mode. For example:

Example:

```
Router# exit
Router>
```

Configuration Examples for MPLS LDP MIB

Enabling the SNMP Agent Examples

The following example shows how to enable an SNMP agent on the host NMS:

```
Router# configure terminal
Router(config)# snmp-server community
```

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

```
Router(config)# snmp-server community public
```

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.

```
Router(config)# snmp-server community comaccess ro 4
```

The following example shows how to enable the session up and session down LDP notifications:

```
Router(config)# snmp-server enable traps mpls ldp session-up
Router(config)# snmp-server enable traps mpls ldp session-down
```

Additional References

Related Documents

Related Topic	Document Title
MPLS LDP configuration tasks	MPLS Label Distribution Protocol (LDP)
MPLS LDP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Multiprotocol Label Switching Command Reference*
SNMP commands	*Cisco IOS Network Management Command Reference*
SNMP configuration	"Configuring SNMP Support" in the *Network Management Configuration Guide*

Standards

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

MIBs

MIBs	MIBs Link
• *MPLS LDP 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

RFCs	Title
RFC 3036	*LDP Specification*

RFCs	Title
RFC 3037	*LDP Applicability*

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 LDP MIB

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 46: Feature Information for MPLS LDP MIB

Feature Name	Releases	Feature Information
MPLS LDP MIB	12.0(11)ST 12.2(2)T 12.0(21)ST 12.2(13)T 12.0(30)S 12.2(27)SBC 12.2(28)SB 12.2(33)SRA 12.2(33)SXH Cisco IOS XE Release 2.1	The MPLS LDP MIB has been implemented to enable standard, SNMP-based network management of the label switching features in Cisco software. In Cisco IOS Release 12.0(11)ST, this feature was introduced to provide SNMP agent support when using the MPLS LDP MIB on Cisco 7200, Cisco 7500, and Cisco 12000 series routers. In Cisco IOS Release 12.2(2)T, this feature was integrated to provide SNMP agent support when using the MPLS LDP MIB on Cisco 7200 and Cisco 7500 series routers. In Cisco IOS Release 12.0(21)ST, the **snmp-server enable traps mpls ldp** command was introduced. The **snmp-server enable traps mpls ldp** command was integrated into Cisco IOS Release 12.2(13)T. This feature was integrated into Cisco IOS Release 12.0(30)S. This feature was integrated into Cisco IOS Release 12.2(27)SBC. This feature was integrated into Cisco IOS Release 12.2(28)SB. This feature was integrated into Cisco IOS Release 12.2(33)SRA. This feature was integrated into Cisco IOS Release 12.2(33)SXH. This feature was integrated into Cisco IOS XE Release 2.1 and implemented on the Cisco ASR 1000 Series Aggregation Services Routers. The following commands were introduced or modified: **snmp-server enable traps mpls ldp**

Feature Name	Releases	Feature Information
MPLS LDP—MIB Notifications	Cisco IOS XE Release 2.1	This feature provides SNMP traps for critical MPLS LDP events. This feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. The following commands were introduced or modified: **snmp-server enable traps mpls ldp**

CHAPTER 7

MPLS Label Distribution Protocol MIB Version 8 Upgrade

The MPLS Label Distribution Protocol (LDP) MIB Version 8 Upgrade feature enhances the LDP MIB to support the Internet Engineering Task Force (IETF) draft Version 8.

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 LDP MIB Version 8 Upgrade

- Simple Network Management Protocol (SNMP) must be installed and enabled on the label switch routers (LSRs).

- Multiprotocol Label Switching (MPLS) must be enabled on the LSRs.
- LDP must be enabled on the LSRs.

Restrictions for MPLS LDP MIB Version 8 Upgrade

This implementation of the MPLS LDP MIB is limited to read-only (RO) permission for MIB objects, except for MIB object *mplsLdpSessionUpDownTrapEnable* , which has been extended to be writable by the SNMP agent.

Setting this object to a value of true enables both the *mplsLdpSessionUp* and *mplsLdpSessionDown* notifications on the LSR; conversely, setting this object to a value of false disables both of these notifications.

For a description of notification events, see the Events Generating MPLS LDP MIB Notifications in MPLS LDP MIB Version 8 Upgrade section.

Most MPLS LDP MIB objects are set up automatically during the LDP peer discovery (hello) process and the subsequent negotiation of parameters and establishment of LDP sessions between the LDP peers.

The following tables are not implemented in this feature:

- mplsLdpEntityFrParmsTable
- mplsLdpEntityConfFrLRTable
- mplsLdpFrameRelaySesTable
- mplsFecTable
- mplsLdpSesInLabelMapTable
- mplsXCsFecsTable
- mplsLdpSesPeerAddrTable

Information About MPLS LDP MIB Version 8 Upgrade

Feature Design of MPLS LDP MIB Version 8 Upgrade

MPLS is a packet forwarding technology that uses a short, fixed-length value called a label in packets to specify the next hop for packet transport through an MPLS network by means of label switch routers (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 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 the two negotiate basic operating procedures. 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. LDP then creates an active LDP session between the two LSRs to effect the exchange of label binding information. When this process is carried to completion with respect to all of the LSRs in an MPLS network, the result is a label-switched path (LSP), which constitutes an end-to-end packet transmission pathway between the communicating network devices.

By means of LDP, LSRs can collect, distribute, and release 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.

The MPLS LDP MIB has been implemented to enable standard, SNMP-based network management of the label switching features in Cisco 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 MPLS LDP MIB.

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

By means of an SNMP agent, you can access MPLS LDP MIB objects using standard SNMP GET operations, and you can use those objects to accomplish a variety of network management tasks. All the objects in the MPLS LDP MIB follow the conventions defined in the IETF draft MIB entitled *draft-ietf-mpls-ldp-mib-08.txt,* which defines network management objects in a structured and standardized manner. This draft MIB is evolving and is soon expected to be a standard. Accordingly, the MPLS LDP MIB will be implemented in such a way that it tracks the evolution of this IETF document.

However, slight differences exist between the IETF draft MIB and the implementation of equivalent Cisco functions. As a result, some minor translations between the MPLS LDP MIB objects and the internal Cisco data structures are needed. Such translations are accomplished by the SNMP agent, which runs in the background on the NMS workstation as a low-priority process.

The extensive Cisco label switching capabilities provide an integrated approach to managing the large volumes of traffic carried by WANs. These capabilities are integrated into the Layer 3 network services, thus optimizing the routing of high-volume traffic through Internet service provider backbones while, at the same time, ensuring the resistance of the network to link or node failures.

The MPLS Label Distribution Protocol MIB Version 8 Upgrade supports the following functions:

- Tag Distribution Protocol (TDP) (This protocol might not be supported in all software releases.)
- Generation 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 CLI commands
- Specification of the name or the IP address of an NMS workstation in the operating environment to which Cisco 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 MIB conforms to Abstract Syntax Notation One (ASN.1), so the MIB forms a highly structured and idealized database of network management objects.

Using any standard SNMP application, you can retrieve and display information from the MPLS LDP MIB by means of standard SNMP GET and GETNEXT operations.

Note Because the MPLS LDP MIB was not given an Internet Assigned Numbers Authority (IANA) experimental object identifier (OID) at the time of its implementation, Cisco chose to implement the MIB under the ciscoExperimental OID number, as follows: ciscoExperimental 1.3.6.1.4.1.9.10 mplsLdpMIB 1.3.6.1.4.1.9.10.65 If the MPLS LDP MIB is assigned an IANA Experimental OID number, Cisco will replace all objects in the MIB under the ciscoExperimental OID and reposition the objects under the IANA Experimental OID.

Enhancements in Version 8 of the MPLS LDP MIB

Version 8 of the MPLS LDP MIB contains the following enhancements:

- TDP support (This protocol might not be supported in all software releases.)
- Upgraded objects
- New indexing that is no longer based on the number of sessions
- Multiple SNMP context support for Virtual Private Networks (VPNs)

Benefits of MPLS LDP MIB Version 8 Upgrade

- Supports TDP and LDP (TDP might not be supported in all software releases.)
- Establishes LDP sessions between peer devices in an MPLS network
- 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/virtual channel identifier (VPI/VCI) pairs to be used in forming labels
- Gathers statistics related to LDP operations, such as error counters.
- 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 the 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 or not)
    - Displaying the label ranges for platform-wide and interface-specific sessions
    - Displaying the ATM parameters.

Description of MPLS LDP MIB Elements for MPLS LDP MIB Version 8 Upgrade

LDP operations related to an MPLS LDP MIB involve the following functional elements:

- LDP entity--Relates 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 under separate headings below.

In effect, the MPLS LDP MIB provides a network management database that supports real-time access to the various MIB objects in the database. This database reflects the current state of MPLS LDP operations in the network. You can access this network management information database by means of standard SNMP commands issued from an NMS in the MPLS LDP operating environment.

The MPLS LDP MIB supports 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).

- 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.

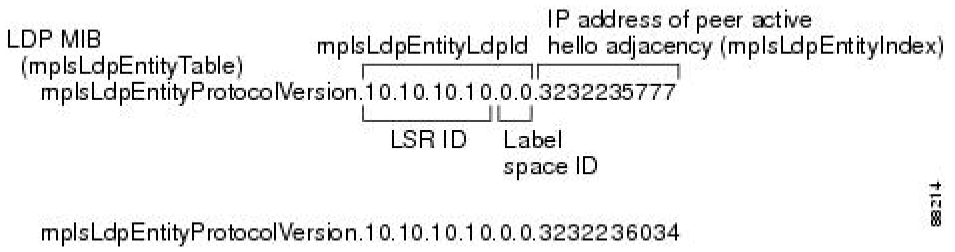
The mplsldpEntityProtocolVersion is a sample object from the mplsLdpEntityTable.

The figure shows the following indexing:

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

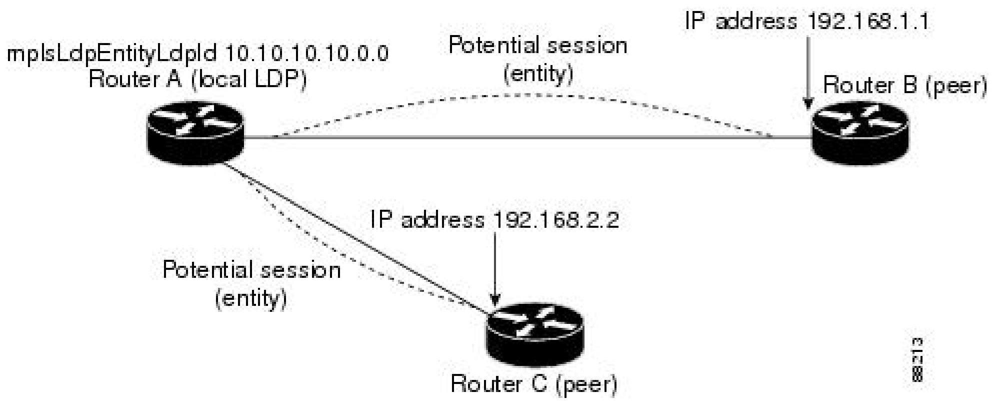
The mplsLdpEntityLdpId or the LDP ID consists of the LSR ID and the label space ID.

- 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.


LDP MIB
(mplsLdpEntityTable)
mplsLdpEntityProtocolVersion.10.10.10.10.0.0.3232235777
mplsLdpEntityLdpId
IP address of peer active hello adjacency (mplsLdpEntityIndex)
LSR ID
Label space ID
mplsLdpEntityProtocolVersion.10.10.10.10.0.0.3232236034
88214

An LDP entity represents a label space that has the potential for a session with an LDP peer. An LDP entity is set up 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.



LDP Sessions and Peers

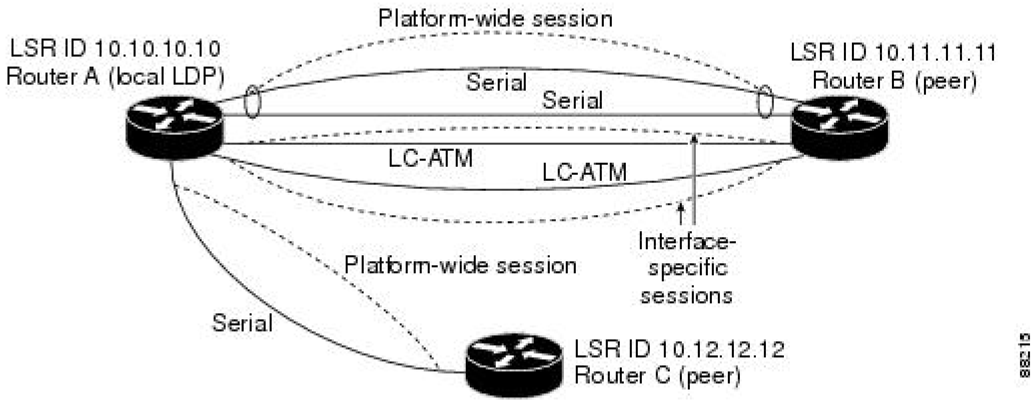
LDP sessions exist between local entities and remote peers for the purpose of distributing label spaces. There is always a one-to-one correspondence between an LDP peer and an LDP session. A single LDP session is an LDP instance that communicates across one or more network links with a single LDP peer.

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/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.



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. There are four mplsLdpSesState sample objects 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.

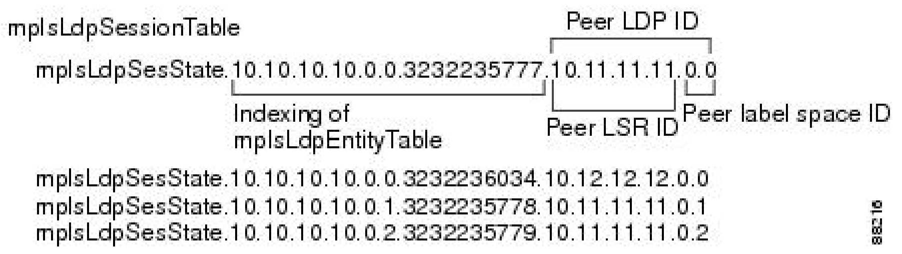
The indexing is based on the entries in the mplsLdpEntityTable. It begins with the indexes of the mplsLdpEntityTable and adds the following:

- 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.


mplsLdpSessionTable
Peer LDP ID
mplsLdpSesState.10.10.10.10.0.0.3232235777.10.11.11.11.0.0
Indexing of mplsLdpEntityTable
Peer LSR ID
Peer label space ID
mplsLdpSesState.10.10.10.10.0.0.3232236034.10.12.12.12.0.0
mplsLdpSesState.10.10.10.10.0.1.3232235778.10.11.11.11.0.1
mplsLdpSesState.10.10.10.10.0.2.3232235779.10.11.11.11.0.2
88216

LDP Hello Adjacencies

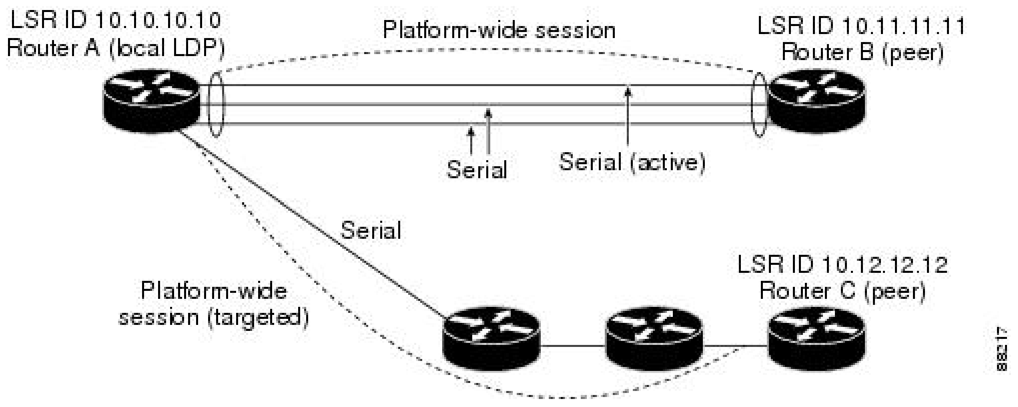
An LDP hello adjacency is a network link between a router and its peers. An LDP hello adjacency enables two adjacent peers to exchange label binding information.

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.

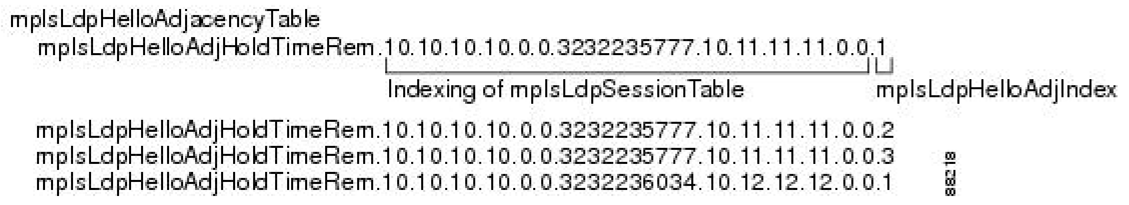
A targeted hello adjacency is not directly connected to its peer and has an unlimited number of hops between itself and its peer. A linked hello adjacency is directly connected between two routers.

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.



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.



```
mplsLdpHelloAdjacencyTable
  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232235777.10.11.11.11.0.0.1

  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232235777.10.11.11.11.0.0.2
  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232235777.10.11.11.11.0.0.3
  mplsLdpHelloAdjHoldTimeRem.10.10.10.10.0.0.3232236034.10.12.12.12.0.0.1
```

Events Generating MPLS LDP MIB Notifications in MPLS LDP MIB Version 8 Upgrade

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

The MPLS LDP MIB objects involved in LDP status transitions and event notifications include 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).
- mplsLdpSessionDown--This message is generated when an LDP session between a local LSR and its adjacent LDP peer is terminated.
- 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.

The value of the path vector limit can range from 0 through 255; a value of 0 indicates that loop detection is off; any value other than zero 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 different 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 set up an LDP session between them, but fail to do so after a specified number of attempts. The default number of attempts is 8. This default value is implemented and cannot be changed.

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. 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/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.

The LSRs whose label ranges do not overlap continue their attempt to create an LDP session between themselves after the eight-retry threshold is exceeded.

In such cases, the LDP threshold exceeded notification alerts the network administrator about a condition in the network that might warrant attention.

RFC 3036, *LDP Specification* , details the incompatibilities that can exist between Cisco routers and/or other vendor LSRs in an MPLS network.

Among such incompatibilities, for example, are the following:

- • Nonoverlapping ATM VPI/VCI ranges (as noted above) or nonoverlapping Frame-Relay DLCI ranges between LSRs attempting to set up an LDP session
- Unsupported label distribution method
- Dissimilar protocol data unit (PDU) sizes
- Dissimilar types of LDP feature support

MIB Tables in MPLS LDP MIB Version 8 Upgrade

Version 8 of the MPLS LDP MIB consists of the following tables:

- mplsLdpEntityTable --Contains entries for every active LDP hello adjacency. 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.

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). Previous implementations of the IETF MPLS-LDP MIB used sessions as the entries in this table. This approach was inadequate because as sessions went down, the entries in the entity table would disappear completely because the agent code could no longer access them. This resulted in the MIB failing to provide information about failed LDP sessions.

Directed adjacencies are also shown in this table. These entries, however, are always up administratively (adminStatus) and operationally (operStatus), because the adjacencies disappear if the directed session fails. Nondirected adjacencies might disappear from the MIB on some occasions, because adjacencies are deleted if the underlying interface becomes operationally down, for example.

- mplsLdpEntityConfGenLRTable --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 mplsLdpEntityConfAtmLRTable instead.) Indexing is the same as it is for the mplsLdpEntityTable, except two indexes have been added, mplsLdpEntityConfGenLRMin and mplsLdpEntityConfGenLRMax. These additional indexes allow more than one label range to be defined. However, in the current Cisco implementation, only one global label range is allowed.
- mplsLdpEntityAtmParmsTable --Contains entries for every LDP-enabled LC-ATM interface. This table is indexed the same as the mplsLdpEntityTable although only LC-ATM interfaces are shown.
- mplsLdpEntityConfAtmLRTable --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,

mplsLdpEntityConfAtmLRMinVpi and mplsLdpEntityConfAtmLRMinVci. These additional indexes allow more than one label range to be defined. However, in the current Cisco implementation, only one label range per LC-ATM interface is allowed.

- mplsLdpEntityStatsTable --Augments the mplsLdpEntityTable and shares the exact same indexing for performing GETand GETNEXT operations. This table shows additional statistics for entities.
- mplsLdpPeerTable --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.
- mplsLdpHelloAdjacencyTable --Contains entries for all 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.
- mplsLdpSessionTable --Augments the mplsLdpPeerTable and shares the same indexing for performing GET and GETNEXT operations. This table shows all sessions.
- mplsLdpAtmSesTable --Contains entries for LC-ATM sessions. Indexing is the same as it is for the mplsLdpPeerTable, except two indexes have been added, mplsLdpSesAtmLRLowerBoundVpi and mplsLdpSesAtmLRLowerBoundVci. These additional indexes allow more than one label range to be defined. However, in the current Cisco implementation, only one label range per LC-ATM interface is allowed.
- mplsLdpSesStatsTable --Augments the mplsLdpPeerTable and shares the exact same indexing for performing GET and GETNEXT operations. This table shows additional statistics for sessions.

mplsLdpEntityTable

The table below lists the mplsLdpEntityTable objects and their descriptions.

Table 47: mplsLdpEntityTable Objects and Descriptions

Object	Description
mplsLdpEntityEntry	Represents an LDP entity, which 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	The administrative status of this LDP entity is always up. If the hello adjacency fails, this entity disappears from the mplsLdpEntityTable.

Object	Description
mplsLdpEntityOperStatus	The operational status of this LDP entity. Values are unknown(0), enabled(1), and disabled(2).
mplsLdpEntityTcpDscPort	The TCP discovery port for LDP or TDP. The default value is 646 (LDP).
mplsLdpEntityUdpDscPort	The UDP discovery port for LDP or TDP. The default value is 646 (LDP).
mplsLdpEntityMaxPduLength	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.
mplsLdpEntityInitSesThreshold	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 CLI.
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 LC-ATM or liberal(2) for all other interfaces.
mplsLdpEntityPVLMisTrapEnable	Indicates whether the mplsLdpPVLMismatch trap should be generated. If the value is enabled(1), the trap is generated. If the value is disabled(2), the trap is not generated. The default is disabled(2). **Note** The mplsLdpPVLMismatch trap is generated only if mplsLdpEntityLabelDistMethod is downstreamOnDemand(1).
mplsLdpEntityPVL	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 mplsLdpEntityPVL object is non-zero only if mplsLdpEntityLabelDistMethod is downstreamOnDemand(1).

Object	Description
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 non-zero only if mplsLdpEntityLabelDistMethod is downstreamOnDemand(1).
mplsLdpEntityTargPeer	If this LDP entity uses a targeted adjacency, this object is set to true(1). The default value is false(2).
mplsLdpEntityTargPeerAddrType	The type of the internetwork layer address used for the extended discovery. This object indicates how the value of mplsLdpEntityTargPeerAddr is to be interpreted.
mplsLdpEntityTargPeerAddr	The value of the internetwork layer address used for the targeted adjacency.
mplsLdpEntityOptionalParameters	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. LC-ATM uses atmParameters(2) to specify that a row in the mplsLdpEntityAtmParmsTable corresponds to this entry. **Note** Frame Relay parameters are not supported.
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.
mplsLdpEntityStorType	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.

mplsLdpEntityConfGenLRTable

The table below lists the mplsLdpEntityConfGenLRTable objects and their descriptions.

Table 48: mplsLdpEntityConfGenLRTable Objects and Descriptions

Object	Description
mplsLdpEntityConfGenLREntry	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 (VPI/VCI pair) and a lower boundary (VPI/VCI pair). The current implementation supports one label range per entity.
mplsLdpEntityConfGenLRMin	The minimum label configured for this range (not accessible).
mplsLdpEntityConfGenLRMax	The maximum label configured for this range (not accessible).
mplsLdpEntityConfGenIfIndxOrZero	This value represents the SNMP IF-MIB index for the platform-wide entity. If the active hello adjacency is targeted, the value is 0.
mplsLdpEntityConfGenLRStorType	The storage type for this entry is a read-only implementation that is always volatile.
mplsLdpEntityConfGenLRRowStatus	This object is a read-only implementation that is always active.

mplsLdpEntityAtmParmsTable

The table below lists the mplsLdpEntityAtmParmsTable objects and their descriptions.

Table 49: mplsLdpEntityAtmParmsTable Objects and Descriptions

Object	Description
mplsLdpEntityAtmParmsEntry	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.

Object	Description
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.
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 VP, 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), the default, and indirect(2).
mplsLdpEntityDefaultControlVpi	The default VPI value for the non-MPLS connection.
mplsLdpEntityDefaultControlVci	The default VCI value for the non-MPLS connection.
mplsLdpEntityUnlabTrafVpi	VPI value of the VCC supporting unlabeled traffic. This non-MPLS connection is used to carry unlabeled (IP) packets.
mplsLdpEntityUnlabTrafVci	VCI value of the VCC supporting unlabeled traffic. This non-MPLS connection is used to carry unlabeled (IP) packets.
mplsLdpEntityAtmStorType	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.

mplsLdpEntityConfAtmLRTable

The table below lists the mplsLdpEntityConfAtmLRTable objects and their descriptions.

Table 50: mplsLdpEntityConfAtmLRTable Objects and Descriptions

Object	Description
mplsLdpEntityConfAtmLREntry	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.
mplsLdpEntityConfAtmLRMinVpi	The minimum VPI number configured for this range (not accessible).
mplsLdpEntityConfAtmLRMinVci	The minimum VCI number configured for this range (not accessible).
mplsLdpEntityConfAtmLRMaxVpi	The maximum VPI number configured for this range (not accessible).
mplsLdpEntityConfAtmLRMaxVci	The maximum VCI number configured for this range (not accessible).
mplsLdpEntityConfAtmLRStorType	The storage type for this entry is a read-only implementation that is always volatile.
mplsLdpEntityConfAtmLRRowStatus	This object is a read-only implementation that is always active.

mplsLdpEntityStatsTable

The table below lists the mplsLdpEntityStatsTable objects and their descriptions.

Table 51: mplsLdpEntityStatsTable Objects and Descriptions

Object	Description
mplsLdpEntityStatsEntry	These entries augment the mplsLdpEntityTable by providing additional information for each entry.
mplsLdpAttemptedSessions	Not supported in this feature.
mplsLdpSesRejectedNoHelloErrors	A count of the session rejected/no hello error notification messages sent or received by this LDP entity.

Object	Description
mplsLdpSesRejectedAdErrors	A count of the session rejected/parameters advertisement mode error notification messages sent or received by this LDP entity.
mplsLdpSesRejectedMaxPduErrors	A count of the session rejected/parameters max PDU length error notification messages sent or received by this LDP entity.
mplsLdpSesRejectedLRErrors	A count of the session rejected/parameters label range notification messages sent or received by this LDP entity.
mplsLdpBadLdpIdentifierErrors	A count of the number of bad LDP identifier fatal errors detected by the session associated with this LDP entity.
mplsLdpBadPduLengthErrors	A count of the number of bad PDU length fatal errors detected by the session associated with this LDP entity.
mplsLdpBadMessageLengthErrors	A count of the number of bad message length fatal errors detected by the session associated with this LDP entity.
mplsLdpBadTlvLengthErrors	A count of the number of bad Type-Length-Value (TLV) length fatal errors detected by the session associated with this LDP entity.
mplsLdpMalformedTlvValueErrors	A count of the number of malformed TLV value fatal errors detected by the session associated with this LDP entity.
mplsLdpKeepAliveTimerExpErrors	A count of the number of session keepalive timer expired errors detected by the session associated with this LDP entity.
mplsLdpShutdownNotifReceived	A count of the number of shutdown notifications received related to the session associated with this LDP entity.
mplsLdpShutdownNotifSent	A count of the number of shutdown notifications sent related to the session associated with this LDP entity.

mplsLdpPeerTable

The table below lists the mplsLdpPeerTable objects and their descriptions.

Table 52: 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).
mplsLdpPeerLoopDetectionForPV	An indication of whether loop detection based on path vectors is disabled or enabled for this peer. For downstream unsolicited distribution (mplsLdpPeerLabelDistMethod is downstreamUnsolicited(2)), this object always has a value of disabled(0) and loop detection is disabled. For downstream-on-demand distribution (mplsLdpPeerLabelDistMethod is downstreamOnDemand(1)), this object has a value of enabled(1), provided that loop detection based on path vectors is enabled.
mplsLdpPeerPVL	If the value of mplsLdpPeerLoopDetectionForPV for this entry is enabled(1), this object represents that path vector limit for this peer. If the value of mplsLdpPeerLoopDetectionForPV for this entry is disabled(0), this value should be 0.

mplsLdpHelloAdjacencyTable

The table below lists the mplsLdpHelloAdjacencyTable objects and their descriptions.

Table 53: 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).

Object	Description
mplsLdpHelloAdjIndex	An identifier for this specific adjacency (not accessible). The active hello adjacency has mplsLdpHelloAdjIndex equal to 1.
mplsLdpHelloAdjHoldTimeRem	The time remaining for this hello adjacency. This interval changes when the next hello message, which corresponds to this hello adjacency, is received.
mplsLdpHelloAdjType	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).

mplsLdpSessionTable

The table below lists the mplsLdpSessionTable objects and their descriptions.

Table 54: 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.
mplsLdpSesState	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)
mplsLdpSesProtocolVersion	The version of the LDP protocol which this session is using. This is the version of the LDP protocol that has been negotiated during session initialization.
mplsLdpSesKeepAliveHoldTimeRem	The keepalive hold time remaining for this session.

Object	Description
mplsLdpSesMaxPduLen	The value of maximum allowable length for LDP PDUs for this session. This value could have been negotiated during the session initialization.
mplsLdpSesDiscontinuityTime	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.

mplsLdpAtmSesTable

The table below lists the mplsLdpAtmSesTable objects and their descriptions.

Table 55: mplsLdpAtmSesTable Objects and Descriptions

Objects	Description
mplsLdpAtmSesEntry	An entry in this table represents information on a single label range intersection between an LDP entity and an LDP peer (not accessible).
mplsLdpAtmSesLRLowerBoundVpi	The minimum VPI number for this range (not accessible).
mplsLdpAtmSesLRLowerBoundVci	The minimum VCI number for this range (not accessible).
mplsLdpAtmSesLRUpperBoundVpi	The maximum VPI number for this range (read-only).
mplsLdpAtmSesLRUpperBoundVci	The maximum VCI number for this range (read-only).

mplsLdpSesStatsTable

The table below lists the mplsLdpSesStatsTable objects and their descriptions.

Table 56: mplsLdpSesStatsTable Objects and Descriptions

Object	Description
mplsLdpSesStatsEntry	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.
mplsLdpSesStatsUnkMesTypeErrors	This object is the count of the number of unknown message type errors detected during this session.
mplsLdpSesStatsUnkTlvErrors	This object is the count of the number of unknown TLV errors detected during this session.

VPN Contexts in MPLS LDP MIB Version 8 Upgrade

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.

This feature adds support for different contexts for different MPLS VPNs. Users of the MIB can view MPLS LDP processes for a given MPLS VPN. The VPN Aware LDP MIB feature does not change the syntax of the IETF MPLS-LDP MIB. It changes the number and types of entries within the tables.

The IETF MPLS-LDP MIB can show information about only one context at a time. You can specify a context, either a global context or an MPLS VPN context, using an SMNP security name.

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

SNMP Context

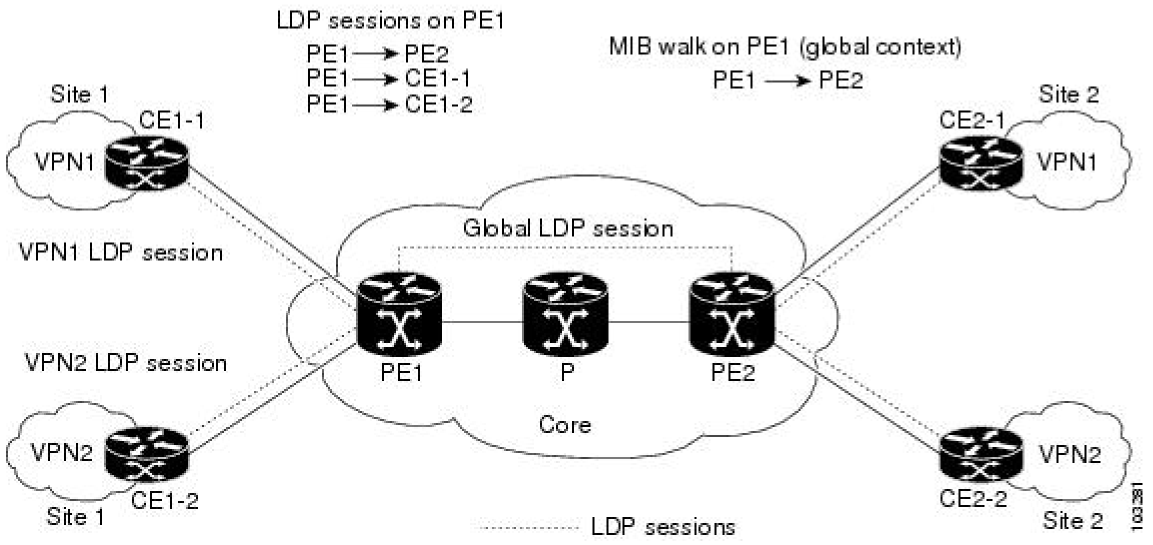
SNMP contexts provide VPN users with a secure way of accessing MIB data. When a VPN is associated with a context, that VPN's specific MIB data exists in that context. Associating a VPN with a context enables service providers to manage networks with multiple VPNs. Creating and associating a context with 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 Aware LDP MIB Sessions

Before the VPN Aware LDP MIB features, an SNMP query to the MPLS LDP MIB returned information about global sessions only. A query did not return information about LDP sessions in a VPN context. The IETF MPLS LDP MIB retrieved information from global routing tables, but did not retrieve information from VPN routing and forwarding instances (VRFs) that store per-VPN routing data. The MPLS LDP MIB looked only at LDP processes in the global context and ignored all other sessions. A query on a VRF returned no information. You can view LDP processes in a VPN context.

The figure below shows a sample MPLS VPN network with the MPLS LDP sessions prior to the implementation of the VPN Aware LDP MIB feature.

Figure 27: MPLS LDP Sessions Setup Before VPN Aware LDP MIB Feature



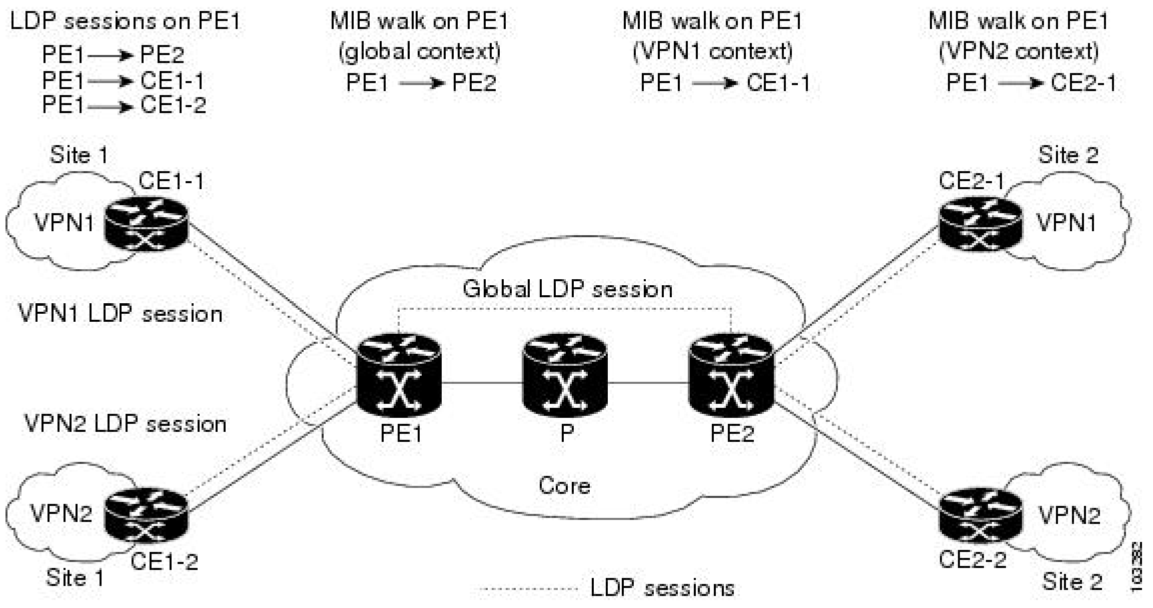
A MIB walk prior to this software release displayed only global session information.

With the VPN Aware LDP MIB enhancement, an SNMP query to the IETF MPLS-LDP-MIB supports 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 update to the LDP MIB also allows you to view 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 as well as 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 IETF MPLS LDP MIB.

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

Figure 28: 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.

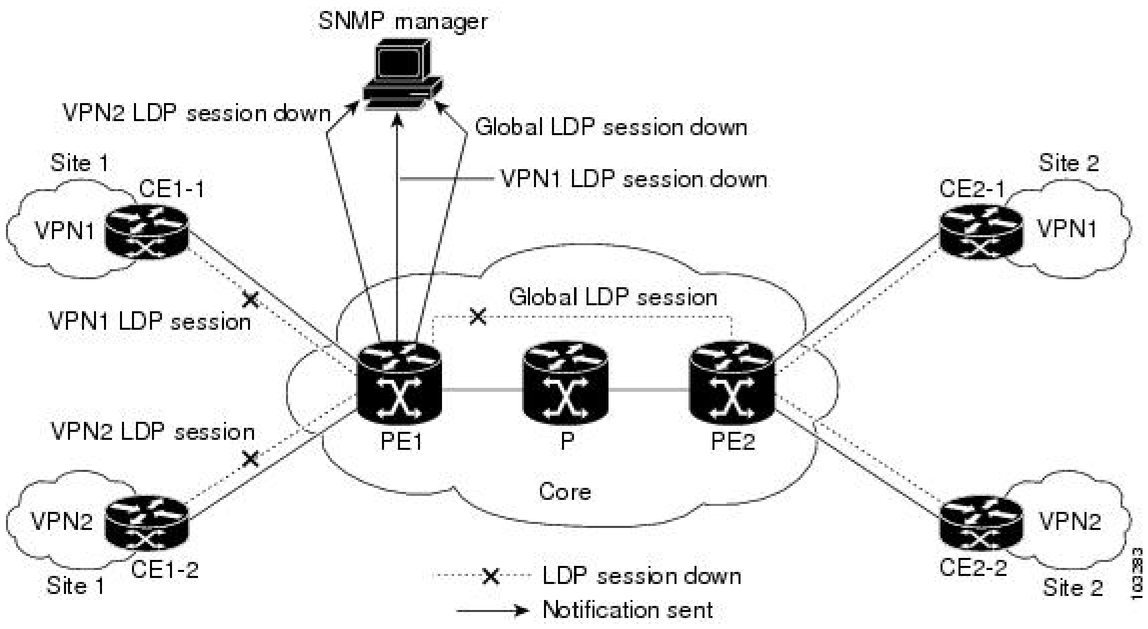
Note To verify LDP session information for a specific VPN, use the **show mpls ldp neighbor vrf** *vpn-name* **detail** command.

VPN Aware LDP MIB Notifications

Before the VPN Aware LDP MIB feature, all notification messages for MPLS LDP sessions were sent to the same designated network management station (NMS) in the network. The notifications were enabled with the **snmp-server enable traps mpls ldp** command.

The figure below shows LDP notifications that were sent before the implementation of the VPN Aware LDP MIB feature.

Figure 29: LDP Notifications Sent Before the VPN Aware LDP MIB Feature



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 object mplsLdpSessionsUpDownEnable (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 PE router:

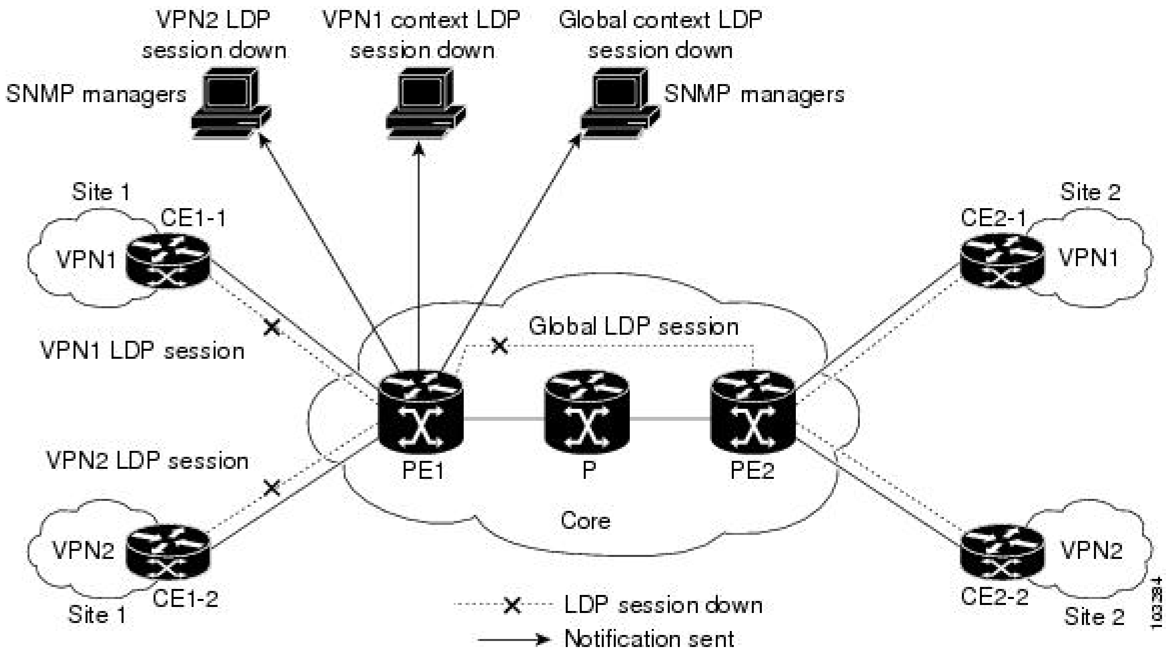
```
Router(config)# snmp-server host host-address traps community mpls-ldp
Router(config)# snmp-server enable traps mpls ldp
```

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

```
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 ldp
```

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

Figure 30: LDP Notifications With the VPN Aware LDP MIB Feature



How to Configure MPLS LDP MIB Version 8 Upgrade

Enabling the SNMP Agent

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **configure terminal**
4. **snmp-server community** *string* [**view** *view-name*] [**ro** *number*]
5. **end**
6. **write memory**
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 of the router so that you can determine if an SNMP agent is already running on the device. 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 desired.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server community** *string* [**view** *view-name*] [**ro** *number*] **Example:** `Router(config)# snmp-server community public ro`	Configures read-only (ro) community strings for the MPLS Label Distribution Protocol (LDP) MIB. • The *string* argument functions like a password, permitting access to SNMP functionality on label switch routers (LSRs) in an MPLS network. • The optional **ro** keyword configures read-only (ro) access to the objects in the MPLS LDP MIB.

Step 5	**end**
	Example:
	`Router(config)# end`
		Exits to privileged EXEC mode.

Step 6	**write memory** **Example:** `Router# write memory`	Writes the modified SNMP configuration into NVRAM of the router, permanently saving the SNMP settings.
Step 7	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration of the router so that you can determine if an SNMP agent is already running on the device. If you see any snmp-server statements, SNMP has been enabled on the router. If any SNMP information is displayed, you can modify the information or change it as desired.

Enabling Distributed Cisco Express Forwarding

Perform this task to enable Cisco Express Forwarding or distributed Cisco Express Forwarding.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip cef distributed**
4. **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	**ip cef distributed** **Example:** `Router(config)# ip cef distributed`	Enables distributed Cisco Express Forwarding.

Step 4	**end**
	
	Example:
	
	`Router(config)# end`
		Exits to privileged EXEC mode.

Enabling MPLS Globally

Perform this task to enable MPLS globally.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mpls ip**
4. **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	**mpls ip** **Example:** `Router(config)# mpls ip`	Enables MPLS forwarding of IPv4 packets along normally routed paths for the platform.

Step 4	**end**
	
	Example:
	`Router(config)# end`
		Exits to privileged EXEC mode.

Enabling LDP Globally

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mpls label protocol** {**ldp** | **tdp**}
4. **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	**mpls label protocol** {**ldp** | **tdp**}
	
	Example:
	
	`Router(config)# mpls label protocol ldp`
		Specifies the platform default label distribution protocol. TDP might not be supported in all software releases.

Step 4	**end**
	
	Example:
	
	`Router(config)# end`
		Exits to privileged EXEC mode.

Enabling MPLS on an Interface

Perform this task to enable MPLS on an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot***/***subslot***/***port* [**.***subinterface-number*]
4. **mpls ip**
5. **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	**interface** *type slot*/*subslot*/*port* [**.***subinterface-number*] **Example:** `Router(config)# interface FastEthernet 1/0/0`	Configures an interface type and enters interface configuration mode.
Step 4	**mpls ip** **Example:** `Router(config-if)# mpls ip`	Enables MPLS forwarding of IPv4 packets along normally routed paths for a particular interface.

Step 5	**end**
	Example:
	`Router(config-if)# end`
		Exits to privileged EXEC mode.

Enabling LDP on an Interface

Perform this task to enable LDP on an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot* / *subslot* / *port* [**.** *subinterface-number*]
4. **mpls label protocol ldp**
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	**interface** *type slot* / *subslot* / *port* [**.** *subinterface-number*] **Example:** `Router(config)# interface FastEthernet 1/0/0`	Configures an interface type and enters interface configuration mode.
Step 4	**mpls label protocol ldp** **Example:** `Router(config-if)# mpls label protocol ldp`	Specifies the label distribution protocol to be used on a given interface.

Step 5	**end**
	
	Example:
	
	`Router(config-if)# end`
		Exits to privileged EXEC mode.

Configuring a VPN Aware LDP MIB

Configuring SNMP Support for a VPN

Perform this task to configure SNMP support for a Virtual Private Network (VPN) or a remote VPN.

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 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* [**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 example.com vrf trap-vrf`	Specifies the recipient of an SNMP notification operation and specifies the Virtual Private Network (VPN) routing and forwarding (VRF) instance table to be used for the sending of SNMP notifications.
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`	Configures a name for the remote SNMP engine on a router.

Step 5	**end**
	
	Example:
	`Router(config)# end`
		Exits to privileged EXEC mode.

Configuring an SNMP Context for a VPN

Perform this task to configure an SNMP context for a VPN. 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 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.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server context** *context-name* **Example:** `Router(config)# snmp-server context context1`	Creates and names an SNMP context.
Step 4	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf vrf1`	Configures a Virtual Private Network (VPN) routing and forwarding instance (VRF) table and enters VRF configuration mode.
Step 5	**rd** *route-distinguisher* **Example:** `Router(config-vrf)# rd 100:120`	Creates a VPN route distinguisher.
Step 6	**context** *context-name* **Example:** `Router(config-vrf)# context context1`	Associates an SNMP context with a particular VRF.
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.

Step 8	**end**
	
	Example:
	
	`Router(config)# end`
		Exits to privileged EXEC mode.

Associating an SNMP VPN Context with SNMPv1 or SNMPv2

Perform this task to associate an SNMP VPN context with SNMPv1 or SNMPv2. 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* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [**notification-type**] [**vrf** *vrf-name*]
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. **exit**

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 user** *username group-name* [**remote** *host* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*]	Configures a new user to an SNMP group.

	Command or Action	Purpose
	Example: `Router(config)# snmp-server user customer1 group1 v1`	
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*] **Example:** `Router(config)# snmp-server group group1 v1 context context1 read view1 write view1 notify view1`	Configures a new SNMP group or a table that maps SNMP users to SNMP views. • Use the **context** *context-name* keyword and argument to associate the specified SNMP group with a configured SNMP context.

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.

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.
Step 7	**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 10.0.0.1 vrf customer1 public udp-port 7002`	Specifies the recipient of an SNMP notification operation.
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-maps community1 context context1 target-list commAVpn`	Associates an SNMP community with an SNMP context, Engine ID, or security name.
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.

	Command or Action	Purpose
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	**exit** **Example:** `Router(config) exit`	Exits to privileged EXEC mode.

Verifying MPLS LDP MIB Version 8 Upgrade

Perform a MIB walk using your SNMP management tool to verify that the MPLS LDP MIB Version 8 Upgrade feature is functioning.

Configuration Examples for MPLS LDP MIB Version 8 Upgrade

MPLS LDP MIB Version 8 Upgrade Examples

The following example shows how to enable an SNMP agent on the host NMS:

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# snmp-server community
```

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

```
Router(config)# snmp-server community public
```

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.

```
Router(config)# snmp-server community comaccess ro 4
```

The following example shows how to enable LDP globally and then on an interface:

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# mpls label protocol ldp
Router(config)# interface FastEthernet1/0/0
Router(config-if)# mpls label protocol ldp
Router(config-if)# 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 Version 8 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 FastEthernet0/3/1
 description Belongs to VPN A
 ip vrf forwarding CustomerA
 ip address 10.0.0.0 255.255.0.0

interface FastEthernet0/3/2
 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 configuration tasks	MPLS Label Distribution Protocol (LDP)

Related Topic	Document Title
A description of SNMP agent support for the MPLS Traffic Engineering MIB (MPLS TE MIB)	MPLS Traffic Engineering (TE) MIB
A description of MPLS differentiated types of service across an MPLS network	MPLS Quality of Service
SNMP commands	*Network Management Command Reference*
SNMP configuration SNMP Support for VPNs	Configuring SNMP Support

Standards

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

MIBs

MIBs	MIBs Link
• MPLS Label Distribution Protocol MIB (draft-ietf-mpls-ldp-mib-08.txt) • SNMP-VACM-MIB The View-based Access Control Model (ACM) MIB for SNMP	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

RFCs	Title
RFC 2233 The LDP implementation supporting the MPLS LDP MIB fully complies with the provisions of Section 10 of RFC 2026, which, in effect, states that the implementation of LDP is recommended for network devices that perform MPLS forwarding along normally routed paths, as determined by destination-based routing protocols.	*Interfaces MIB*

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 LDP MIB Version 8 Upgrade

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 57: Feature Information for MPLS LDP MIB Version 8 Upgrade

Feature Name	Releases	Feature Information
MPLS LDP MIB Version 08 Upgrade	12.0(11)ST 12.2(2)T 12.0(21)ST 12.0(22)S 12.0(24)S 12.2(18)S 12.2(33)SRB 12.2(33)SB Cisco IOS XE Release 2.1	

Feature Name	Releases	Feature Information
		The MPLS Label Distribution Protocol (LDP) MIB Version 8 Upgrade feature enhances the LDP MIB to support the Internet Engineering Task Force (IETF) draft Version 8. In Cisco IOS Release 12.0(11)ST, this feature was introduced to provide SNMP agent support for the MPLS LDP MIB on the Cisco 7200, Cisco 7500, and Cisco 12000 series routers. In Cisco IOS Release 12.2(2)T, this feature was added to this release to provide SNMP agent support for the MPLS LDP MIB on Cisco 7200 and Cisco 7500 series routers. In Cisco IOS Release 12.0(21)ST, this feature was added to this release to provide SNMP agent and LDP notification support for the MPLS LDP MIB on Cisco 7200, Cisco 7500, and Cisco 12000 series Internet routers. In Cisco IOS Release 12.0(22)S, Version 1 was integrated into Cisco IOS Release 12.0(22)S. In Cisco IOS Release 12.0(24)S, this feature was upgraded to Version 8 in Cisco IOS Release 12.0(24)S. This feature was integrated into Cisco IOS Release 12.2(18)S. In Cisco IOS Release 12.2(33)SRB, this MIB was deprecated and replaced by MPLS-LDP-STD-MIB (RVC 3815). In Cisco IOS Release 12.2(33)SB, this MIB was deprecated and replaced by MPLS-LDP-STD-MIB (RVC 3815). This feature was integrated into Cisco IOS XE Release 2.1 and implemented on Cisco ASR 1000

Feature Name	Releases	Feature Information
		Series Aggregation Services Routers.
		The following commands were introduced or modified: **context**, **show mpls ldp neighbor**, **snmp mib community-map**, **snmp mib target list**, **snmp-server community**, **snmp-server context**, **snmp-server enable traps** (MPLS), **snmp-server group**, **snmp-server host**, **snmp-server trap authentication vrf**.
MPLS VPN-VPN Aware LDP MIB	12.0(27)S 12.2(28)SB 12.2(33)SRA 12.2(33)SXH 12.4(20)T	The MPLS VPN-VPN Aware LDP MIB allows you to enter LDP queries on any VRF and on the core (global context). In Cisco IOS Release 12.0(27)S, support for the MPLS VPN-VPN Aware LDP MIB feature was added. In Cisco IOS Release 12.2(28)SB, this feature was integrated. In Cisco IOS Release 12.2(33)SRA, this feature was integrated. In Cisco IOS Release 12.2(33)SXH, this feature was integrated. In Cisco IOS Release 12.4(20)T, this feature was integrated.

Glossary

ATM -- Asynchronous Transfer Mode. The international standard for cell relay in which multiple service types (such as voice, video, or data) are conveyed in fixed-length (53-byte) cells. Fixed-length cells allow cell processing to occur in hardware, thereby reducing transit delays. ATM is designed to take advantage of high-speed transmission media, such as E3, SONET, and T3 .

downstream-on-demand distribution--A label distribution method in which a downstream label switch router (LSR) sends a binding upstream only if the upstream LSR requests it.

downstream unsolicited distribution--A label distribution method in which labels are dispersed if a downstream label switch router (LSR) needs to establish a new binding with its neighboring upstream LSR.

For example, an edge LSR might enable a new interface with another subnet. The LSR then announces to the upstream router a binding to reach this network.

informs --A type of notification message that is more reliable than a conventional trap notification message, because the informs message notification requires acknowledgment, but a trap notification does not.

label --A short, fixed-length data identifier that tells switching nodes how to forward data (packets or cells).

label distribution--The techniques and processes that are used by label switch routers (LSRs) to exchange label binding information for supporting hop-by-hop forwarding along normally routed paths.

LDP --Label Distribution Protocol. The protocol that supports Multiprotocol Label Switching (MPLS) hop-by-hop forwarding and the distribution of bindings between labels and network prefixes.

LSP --label switched path. A configured connection between two label switch routers (LSRs) in which label-switching techniques are used for packet forwarding; also a specific path through an Multiprotocol Label Switching (MPLS) network.

LSR --label switch router. A Multiprotocol Label Switching (MPLS) node that can forward native Layer 3 packets. The LSR forwards a packet based on the value of a label attached to the packet.

MIB --Management Information Base. A database of network management information that is used and maintained by a network management protocol such as Simple Network Management Protocol (SNMP). The value of a MIB object can be changed or retrieved by the use 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.

MPLS --Multiprotocol Label Switching. A switching method for the forwarding of IP traffic through the use of a label. This label instructs the routers and the switches in the network where to forward the packets based on preestablished IP routing information.

MPLS label distribution--A constraint-based routing algorithm for routing label-switched path (LSP) tunnels.

NMS --network management station. A powerful, well-equipped computer (typically an engineering workstation) that is used by a network administrator to communicate with other devices in the network. An NMS is typically used to manage network resources, gather statistics, and perform a variety of network administration and configuration tasks. In the context of Simple Network Management Protocol (SNMP), an NMS is a device that performs SNMP queries to the SNMP agent of a managed device to retrieve or modify information.

notification --A message sent by a Simple Network Management Protocol (SNMP) agent to a network management station, console, or terminal to indicate that a significant network event has occurred. See also trap.

RSVP --Resource Reservation Protocol. A protocol that supports the reservation of resources across an IP network. Applications running on IP end systems can use RSVP to indicate to other nodes the nature of the packet streams they want to receive by specifying such items as bandwidth, jitter, and maximum burst.

RTR --Response Time Reporter. A tool that allows you to monitor network performance, network resources, and applications by measuring response times and availability.

SNMP --Simple Network Management Protocol. A network management protocol used almost exclusively in TCP/IP networks. SNMP enables a user to monitor and control network devices, manage configurations, collect statistics, monitor performance, and ensure network security.

SNMP communities--Authentication scheme that enables an intelligent network device to validate SNMP requests.

SNMPv2c --Version 2c of the Simple Network Management Protocol. SNMPv2c supports centralized as well as distributed network management strategies and includes improvements in the Structure of Management Information (SMI), protocol operations, management architecture, and security.

SNMPv3 --Version 3 of the Simple Network Management Protocol. Interoperable standards-based protocol for network management. SNMPv3 provides secure access to devices by a combination of authenticating and encrypting packets over the network.

TLV --Type-Length-Value. A mechanism used by several routing protocols to carry a variety of attributes. Cisco Discovery Protocol (CDP), Label Discovery Protocol (LDP), and Border Gateway Protocol (BGP) are examples of protocols that use TLVs. BGP uses TLVs to carry attributes such as Network Layer Reachability Information (NLRI), Multiple Exit Discriminator (MED), and local preference.

trap --A message sent by a Simple Network Management Protocol (SNMP) agent to a network management station, console, or terminal to indicate that a significant network event has occurred. Traps (notifications) are less reliable than inform requests, because the receiver of the trap does not send an acknowledgment of receipt; furthermore, the sender of the trap cannot determine if the trap was received. See also notification.

VCC --virtual channel connection. A logical circuit, made up of virtual channel links (VCLs), that carries data between two endpoints in an ATM network. Sometimes called a virtual circuit connection.

VCI --virtual channel identifier. A 16-bit field in the header of an ATM cell. The VCI, together with the virtual path identifier (VPI), is used to identify the next network virtual channel link (VCL) as the cell passes through a series of ATM switches on its way to its final destination.

VCL --virtual channel link. The logical connection that exists between two adjacent switches in an ATM network.

VPI --virtual path identifier. An 8-bit field in the header of an ATM cell. The VPI, together with the virtual channel identifier (VCI), is used to identify the next network virtual channel link (VCL) as the cell passes through a series of ATM switches on its way to its final destination.

VPN --Virtual Private Network. A network that enables IP traffic to use tunneling to travel securely over a public TCP/IP network.

VRF --VPN routing and forwarding instance. A VRF consists of an IP routing table, a derived forwarding table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols that determine what goes into the forwarding table. In general, a VRF includes the routing information that defines a customer VPN site that is attached to a PE router.

CHAPTER 8

MPLS Traffic Engineering MIB

The MPLS Traffic Engineering MIB enables Simple Network Management Protocol (SNMP) agent support in Cisco software for Multiprotocol Label Switching (MPLS) traffic engineering (TE) management, as implemented in the MPLS Traffic Engineering MIB (MPLS TE MIB). The SNMP agent code operating with the MPLS TE MIB enables a standardized, SNMP-based approach to be used in managing the MPLS TE features in Cisco software.

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.

Restrictions for the MPLS Traffic Engineering MIB

- Supports read-only (RO) permission for MIB objects.

- Contains no configuration support by means of SET functions, except for the mplsTunnelTrapEnable object (which has been made writable). Accordingly, the MPLS TE MIB contains indexing support for the Interfaces MIB.
- Supports only SNMP GET, GETNEXT, and GETBULK retrieval functions, except in the case of the mplsTunnelTrapEnable object (which has been made writable by means of SET functions).
- Contains no support for Guaranteed Bandwidth Traffic Engineering (GBTE) or Auto Bandwidth features.

Information About the MPLS Traffic Engineering MIB

MPLS Traffic Engineering MIB Cisco Implementation

The MPLS TE MIB is based on the Internet Engineering Task Force (IETF) draft MIB entitled *draft-ietf-mpls-te-mib-05.txt* which includes objects describing features that support MPLS TE.

Slight differences between the IETF draft MIB and the implementation of the TE capabilities within Cisco software require some minor translations between the MPLS TE MIB and the internal data structures of Cisco software. These translations are made by the SNMP agent code that is installed and operating on various hosts within the network. This SNMP agent code, running in the background as a low priority process, provides a management interface to Cisco software.

The SNMP objects defined in the MPLS TE MIB can be displayed by any standard SNMP utility. All MPLS TE MIB objects are based on the IETF draft MIB; thus, no specific Cisco SNMP application is required to support the functions and operations pertaining to the MPLS TE MIB.

MPLS Traffic Engineering Overview

MPLS TE capabilities in Cisco software enable an MPLS backbone to replicate and expand upon the TE capabilities of Layer 2 ATM and Frame Relay networks.

TE capabilities are essential to effective management of service provider and Internet service provider (ISP) backbones. Such backbones must support high transmission capacities, and the networks incorporating backbones must be extremely resilient to link or node failures.

The MPLS TE facilities built into Cisco software provide a feature-rich, integrated approach to managing the large volumes of traffic that typically flow through WANs. The MPLS TE facilities are integrated into Layer 3 network services, thereby optimizing the routing of IP traffic in the face of constraints imposed by existing backbone transmission capacities and network topologies.

Capabilities Supported by the MPLS Traffic Engineering MIB

- The ability to generate and queue notification messages that signal changes in the operational status of MPLS TE tunnels.
- Extensions to existing SNMP commands that provide the ability to enable, disable, and configure notification messages for MPLS TE tunnels.
- The ability to specify the name or the IP address of a network management station (NMS) in the operating environment to which notification messages are to be sent.

- The ability to write notification configurations into nonvolatile memory.

Notification Generation Events

When MPLS TE notifications are enabled (see the **snmp-server enable traps mpls** command), notification messages relating to specific events within Cisco software are generated and sent to a specified NMS in the network.

For example, an mplsTunnelUp notification is sent to an NMS when an MPLS TE tunnel is configured and the tunnel transitions from an operationally "down" state to an "up" state.

Conversely, an mplsTunnelDown notification is generated and sent to an NMS when an MPLS TE tunnel transitions from an operationally "up" state to a "down" state.

An mplstunnelRerouted notification is sent to the NMS under the following conditions:

- The signaling path of an existing MPLS TE tunnel fails for some reason and a new path option is signaled and placed into effect (that is, the tunnel is rerouted).
- The signaling path of an existing MPLS TE tunnel is fully operational, but a better path option can be signaled and placed into effect (that is, the tunnel can be reoptimized). This reoptimazation can be triggered by:
    - A timer
    - The issuance of an **mpls traffic-eng reoptimize** command
    - A configuration change that requires the resignaling of a tunnel

The mplsTunnelReoptimized notification is not generated when an MPLS traffic engineering tunnel is reoptimized. However, an mplsTunnelReroute notification is generated. Thus, at the NMS, you cannot distinguish between a tunnel reoptimization event and tunnel reroute event.

Path options are configurable parameters that you can use to specify the order of priority for establishing a new tunnel path. For example, you can create a tunnel head configuration and define any one of many path options numbered 1 through n, with "1" being the highest priority option and "n" being an unlimited number of lower priority path options. Thus, there is no limit to the number of path options that you can specify in this manner.

Notification Implementation

When an MPLS TE tunnel interface (or any other device interface, such as an FastEthernet or Packet over SONET (POS) interface) transitions between an up and down state, an Interfaces MIB (ifMIB) link notification is generated. When such a notification occurs in an MPLS TE MIB environment, the interface is checked by software to determine if the notification is associated with an MPLS TE tunnel. If so, the interfaces MIB link notification is interlinked with the appropriate mplsTunnelUp or mplsTunnelDown notification to provide notification to the NMS regarding the operational event occurring on the tunnel interface. Hence, the generation of an Interfaces MIB link notification pertaining to an MPLS traffic engineering tunnel interface begets an appropriate mplsTunnelUp or mplsTunnelDown notification that is transmitted to the specified NMS.

An mplsTunnelRerouted notification is generated whenever the signaling path for an MPLS TE tunnel changes. However, software intelligence in the MPLS TE MIB prevents the reroute notification from being sent to the NMS when a TE tunnel transitions between an up or down state during an administrative or operational status check of the tunnel. Either an up or down notification or a reroute notification can be sent in this instance, but not both. This action prevents unnecessary traffic on the network.

Benefits of the MPLS Traffic Engineering MIB

- Provides a standards-based SNMP interface for retrieving information about MPLS TE.
- Provides information about the traffic flows on MPLS TE tunnels.
- Presents MPLS TE tunnel routes, including the configured route, the Interior Gateway Protocol (IGP) calculated route, and the actual route traversed.
- Provides information, in conjunction with the Interfaces MIB, about how a tunnel was rerouted in the event of a link failure.
- Provides information about the configured resources used for an MPLS TE tunnel.
- Supports the generation and queueing of notifications that call attention to major changes in the operational status of MPLS TE tunnels;
- Forwards notification messages to a designated NMS for evaluation or action by network administrators.

MPLS Traffic Engineering MIB Layer Structure

The SNMP agent code supporting the MPLS TE MIB follows the existing model for such code in Cisco software and is, in part, generated by the Cisco tool set, based on the MIB source code.

The SNMP agent code, which has a layered structure similar to that of the MIB support code in Cisco software, consists of four layers:

- Platform independent layer--This layer is generated primarily by the Cisco MIB development tool set and incorporates platform and implementation independent functions. The Cisco MIB development tool set creates a standard set of files associated with a MIB.
- Application interface layer--The functions, names, and template code for MIB objects in this layer are also generated by the Cisco MIB development tool set.
- Application specific layer--This layer provides an interface between the application interface layer and the application program interface (API) and data structures layer and performs tasks needed to retrieve required information from Cisco software, such as searching through data structures.
- API and data structures layer--This layer contains the data structures or APIs within Cisco software that are retrieved or called in order to set or retrieve SNMP management information.

Features and Technologies Related to MPLS Traffic Engineering MIB

The MPLS TE MIB feature is used with the following features and technologies:

- Standards-based SNMP network management application
- MPLS
- MPLS TE

- MPLS label switching router MIB (MPLS-LSR-MIB)

Supported Objects in the MPLS Traffic Engineering MIB

The MPLS TE MIB contains numerous tables and object definitions that provide read-only SNMP management support for the MPLS TE features in Cisco software. The MPLS TE MIB conforms to Abstract Syntax Notation One (ASN.1), thus reflecting an idealized MPLS TE database.

Using any standard SNMP network management application, you can retrieve and display information from the MPLS TE MIB by using GET operations; similarly, you can traverse information in the MIB database for display by using GETNEXT operations.

The MPLS TE MIB tables and objects supported in Cisco software follow. Important MIB tables (those highlighted in bold type) are described briefly in accompanying text.

- mplsTunnelConfigured--Total number of tunnel configurations that are defined on this node.
- mplsTunnelActive--Total number of label switched paths (LSPs) that are defined on this node.
- mplsTunnelTEDistProto--The IGP distribution protocol in use.
- mplsTunnelMaxHops--The maximum number of hops any given tunnel can utilize.
- mplsTunnelIndexNext--Unsupported; set to 0.
- mplsTunnelTable--Entries in this table with an instance of 0 and a source address of 0 represent tunnel head configurations. All other entries in this table represent instances of LSPs, both signaled and standby. If a tunnel instance is signaled, its operating status (operStatus) is set to “up” (1) and its instance corresponds to an active LSP.

Tunnel configurations exist only on the tunnel head where the tunnel interface is defined. LSPs traverse the network and involve tunnel heads, tunnel midpoints, and tunnel tails.

Pointers in the tunnel table refer to corresponding entries in other MIB tables. By using these pointers, you can find an entry in the mplsTunnelTable and follow a pointer to other tables for additional information. The pointers are the following: mplsTunnelResourcePointer, mplsTunnelHopTableIndex, mplsTunnelARHopTableIndex, and mplsTunnelCHopTableIndex.

The tunnel table is indexed by tunnel ID, tunnel instance, tunnel source address, and tunnel destination address. The description of each entry has an alphabetic suffix (a) for tunnel head configurations only, (b) for LSPs only, or (c) for both tunnel head configurations and LSPs , if appropriate, to indicate the applicability of the entry.

Following is a list and description of each entry.

- • mplsTunnelIndex--Same as tunnel ID (c).
  - mplsTunnelInstance--Tunnel instance of the LSP; 0 for head configurations (b).
  - mplsTunnelIngressLSRId--Source IP address of the LSP; 0 for head configurations (b).
  - mplsTunnelEgressLSRId--Destination IP address of the tunnel (c).
  - mplsTunnelName--Command name for the tunnel interfaces (a).
  - mplsTunnelDescr--Descriptive name for tunnel configurations and LSPs (c).
  - mplsTunnelIsIf--Indicator of whether the entry represents an interface (c).

- mplsTunnelIfIndex--Index of the tunnel interface within the ifMIB (a).
- mplsTunnelXCPointer--(For midpoints only - no tails) Pointer for the LSP within the mplsXCTable of the MPLS LSR MIB (b).
- mplsTunnelSignallingProto--Signaling protocol used by tunnels (c).
- mplsTunnelSetupPrio--Setup priority of the tunnel (c).
- mplsTunnelHoldingPrio--Holding priority of the tunnel (c).
- mplsTunnelSessionAttributes--Session attributes (c).
- mplsTunnelOwner--Tunnel owner (c).
- mplsTunnelLocalProtectInUse--Not implemented (c).
- mplsTunnelResourcePointer--Pointer into the Resource Table (b).
- mplsTunnelInstancePriority--Not implemented (b).
- mplsTunnelHopTableIndex--Index into the Hop Table (a).
- mplsTunnelARHopTableIndex--Index into the AR Hop Table (b).
- mplsTunnelCHopTableIndex--Index into the C Hop Table (b).
- mplsTunnelPrimaryTimeUp--Amount of time, in seconds, that the current path has been up (a).
- mplsTunnelPathChanges--Number of times a tunnel has been resignalled (a).
- mplsTunnelLastPathChange--Amount of time, in seconds, since the last path resignaling occurred (a).
- mplsTunnelCreationTime--Time stamp when the tunnel was created (a).
- mplsTunnelStateTransitions--Number of times the tunnel has changed state (a).
- mplsTunnelIncludeAnyAffinity--Not implemented (a).
- mplsTunnelIncludeAllAffinity--Attribute bits that must be set for the tunnel to traverse a link (a).
- mplsTunnelExcludeAllAffinity--Attribute bits that must *not* be set for the tunnel to traverse a link (a).
- mplsTunnelPathInUse--Path option number being used for the tunnel's path. If no path option is active, this object will be 0 (a).
- mplsTunnelRole--Role of the tunnel on the router; that is, head, midpoint, or tail (c).
- mplsTunneltotalUptime--Amount of time, in seconds, that the tunnel has been operationally up (a).
- mplsTunnelInstanceUptime--Not implemented (b).
- mplsTunnelAdminStatus--Administrative status of a tunnel (c).
- mplsTunnelOperStatus--Actual operating status of a tunnel (c).
- mplsTunnelRowStatus--This object is used in conjunction with configuring a new tunnel. This object will always be seen as "active" (a).
- mplsTunnelStorageType--Storage type of a tunnel entry (c).

- mplsTunnelHopListIndexNext--Next valid index to use as an index in the mplsTunnelHopTable.
- **mplsTunnelHopTable** --Entries in this table exist only for tunnel configurations and correspond to the path options defined for the tunnel. Two types of path options exist: *explicit* and *dynamic* . This table shows all hops listed in the explicit path options, while showing only the destination hop for dynamic path options. The tunnel hop table is indexed by tunnel ID, path option, and hop number.

Following is a list and description of each table entry.

-
    - mplsTunnelHopListIndex--Primary index into the table.
    - mplsTunnelHopIndex--Secondary index into the table.
    - mplsTunnelHopAddrType--Indicates if the address of this hop is the type IPv4 or IPv6.
    - mplsTunnelHopIpv4Addr--The IPv4 address of this hop.
    - mplsTunnelHopIpv4PrefixLen--The prefix length of the IPv4 address.
    - mplsTunnelHopIpv6Addr--The IPv6 address of this hop.
    - mplsTunnelHopIpv6PrefixLen--The prefix length of the IPv6 address.
    - mplsTunnelHopAsNumber--This object will contain 0 or the autonomous system number of the hop, depending on the value of mplsTunnelHopAddrType.
    - mplsTunnelHopLspId--This object will contain 0 or the LSPID of the tunnel, depending on the value of mplsTunnelHopAddrType.
    - mplsTunnelHopType--Denotes whether this tunnel hop is routed in a strict or loose fashion.
    - mplsTunnelHopRowStatus--This object is used in conjunction with the configuring of a new row in the table.
    - mplsTunnelHopStorageType--The storage type of this MIB object.

- mplsTunnelResourceIndexNext--This object contains the next appropriate value to be used for mplsTunnelResourceIndex when creating entries in the mplsTunnelResourceTable
- **mplsTunnelResourceTable** --Entries in this table correspond to the "Tspec" information displayed when you execute the **show mpls traff9c-eng tunnels** command. These entries exist only for LSPs.

The tunnel resource table is indexed by address and hop number. Following the mplsTunnelResourcePointer pointer from the tunnel table is the best way to retrieve information from this table.

Following is a list and description of each table entry.

-
    - mplsTunnelResourceIndex--The primary index into this table.
    - mplsTunnelResourceMaxRate--The maximum rate, in bits per second, supported by this tunnel.
    - mplsTunnelResourceMeanRate--The mean rate, in bits per second, supported by this tunnel.
    - mplsTunnelResourceMaxBurstSize--The maximum burst size, in bytes, allowed by this tunnel.
    - mplsTunnelResourceRowStatus--This object is used in conjunction with the configuration of a new row in the table.
    - mplsTunnelResourceStorageType--The storage type of this MIB object.

- **mplsTunnelARHopTable** --Entries in this table correspond to the actual route taken by the tunnel, and whose route was successfully signaled by the network. The hops present in this table correspond to those present in the record route object (RRO) in Resource Reservation Protocol (RSVP). You can also display the information in this table by executing the **show mpls traff9c-eng tunnels** command.

The actual route hop table is indexed by address and hop number. Following the mplsTunnelARHopTableIndex pointer from the tunnel table is the best way to retrieve information from this table.

Following is a list and description of each table entry:

-
    - mplsTunnelARHopListIndex--The primary index into this table.
    - mplsTunnelARHopIndex--The secondary index into this table.
    - mplsTunnelARHopIpv4Addr--The IPv4 address of this hop.
    - mplsTunnelARHopIpv4PrefixLen--The prefix length of the IPv4 address.
    - mplsTunnelARHopIpv6Addr--The IPv6 address of this hop.
    - mplsTunnelARHopIpv6PrefixLen--The prefix length of the IPv6 address.
    - mplsTunnelARHopAsNumber--This object will contain 0 or the AS number of the hop, depending on the value of mplsTunnelARHopAddrType.
    - mplsTunnelARHopAddrType--The type of address for this MIB entry, either IPv4 or IPv6.
    - mplsTunnelARHopType--Denotes whether this tunnel hop is routed in a strict or loose manner.

- **mplsTunnelCHopTable** --Entries in this table correspond to the explicit route object (ERO) in RSVP, which is used to signal the LSP. The list of hops in this table will contain those hops that are computed by the constraint-based shortest path first (SPF) algorithm. In those cases where “loose” hops are specified for the tunnel, this table will contain the hops that are “filled-in” between the loose hops to complete the path. If you specify a complete explicit path, the computed hop table matches your specified path.

The computed hop table is indexed by address and hop number. Following the mplsTunnelCHopTableIndex pointer from the tunnel table is the best way to retrieve information from this table.

Following is a list and description of each table entry:

-
    - mplsTunnelCHopListIndex--The primary index into this table.
    - mplsTunnelCHopIndex--The secondary index into this table.
    - mplsTunnelCHopAddrType--Indicates if the address of this hop is the type IPv4 or IPv6.
    - mplsTunnelCHopIpv4Addr--The IPv4 address of this hop.
    - mplsTunnelCHopIpv4PrefixLen--The prefix length of the IPv4 address.
    - mplsTunnelCHopIpv6Addr--The IPv6 address of this hop.
    - mplsTunnelCHopIpv6PrefixLen--The prefix length of the IPv6 address.
    - mplsTunnelCHopAsNumber--This object will contain 0 or the autonomous system number of the hop, depending on the value of mplsTunnelHopAddrType.
    - mplsTunnelCHopType--Denotes whether this tunnel hop is routed in a strict or loose way.

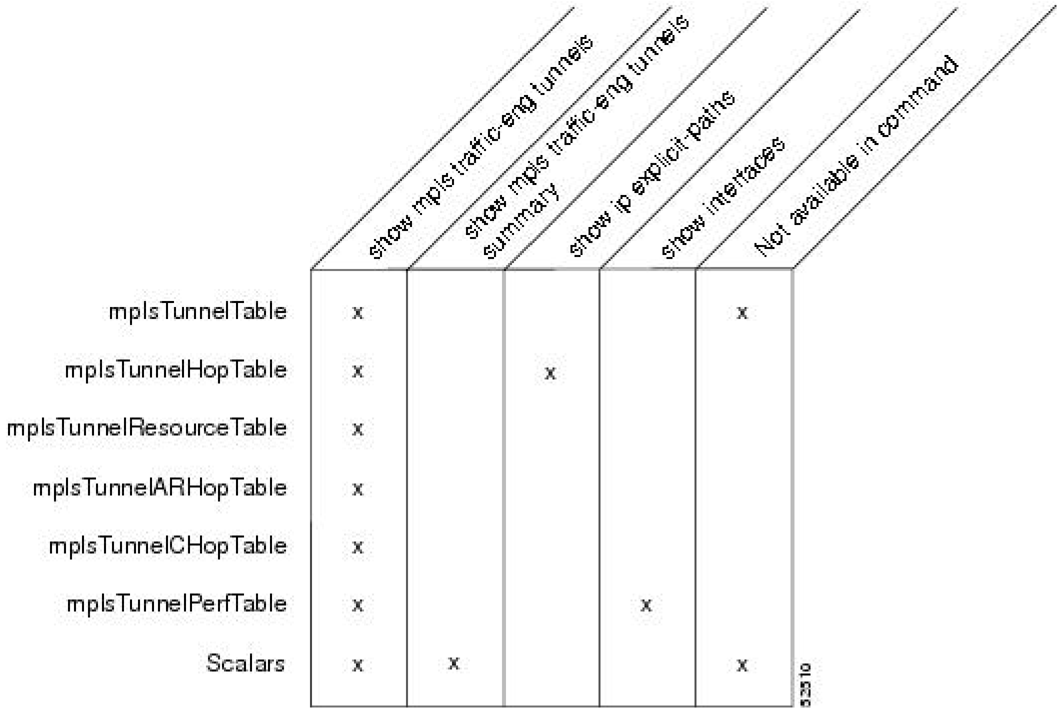
- **mplsTunnelPerfTable** --The tunnel performance table, which augments the **mplsTunnelTable**, provides packet and byte counters for each tunnel. This table contains the following packet and byte counters:
  - mplsTunnelPerfPackets--This packet counter works only for tunnel heads.
  - mplsTunnelPerfHCPackets--This packet counter works only for tunnel heads.
  - mplsTunnelPerfErrors--This packet counter works only for tunnel heads.
  - mplsTunnelPerfBytes--This byte counter works for tunnel heads and tunnel midpoints, but not for tunnel tails.
  - mplsTunnelPerfHCBytes--This byte counter works for tunnel heads and tunnel midpoints, but not for tunnel tails.
- mplsTunnelTrapEnable--The object type *mplsTunnelTrapEnable* is enhanced to be writable. Accordingly, if this object type is set to "TRUE," the following notifications are enabled, thus giving you the ability to monitor changes in the operational status of MPLS TE tunnels:
  - mplsTunnelUp
  - mplsTunnelDown
  - mplsTunnelRerouted

If the *mplsTunnelTrapEnable* object is set to "FALSE," such operational status notifications are not generated. These notification functions are based on the definitions (mplsTeNotifications) contained in the IETF draft document entitled *draft-ietf-mpls-te-mib-05.txt* .

CLI Access to MPLS Traffic Engineering MIB Information

The figure below shows commands that you can use to retrieve information from specific tables in the MPLS TE MIB. As noted in this figure, some information in the MPLS TE MIB is not retrievable by commands.

Figure 31: Commands for Retrieving MPLS TE MIB Information



	show mpls traffic-eng tunnels	show mpls traffic-eng tunnels summary	show ip explicit-paths	show interfaces	Not available in command
mplsTunnelTable	x				x
mplsTunnelHopTable	x		x		
mplsTunnelResourceTable	x				
mplsTunnelARHopTable	x				
mplsTunnelCHopTable	x				
mplsTunnelPerfTable	x			x	
Scalars	x	x			x

Retrieving Information from the MPLS Traffic Engineering MIB

This section describes how to efficiently retrieve information about TE tunnels. Such information can be useful in large networks that contain many TE tunnels.

Traverse across a single column of the mplsTunnelTable, such as mplsTunnelName. This action provides the indexes of every tunnel configuration, and any LSPs involving the host router. Using these indexes, you can perform a GET operation to retrieve information from any column and row of the mplsTunnelTable.

The mplsTunnelTable provides pointers to other tables for each tunnel. The column mplsTunnelResourcePointer, for example, provides an object ID (OID) that you can use to access resource allocation information in the mplsTunnelResourceTable. The columns mplsTunnelHopTableIndex, mplsTunnelARHopTableIndex, and mplsTunnelCHopTableIndex provide the primary index into the mplsTunnelHopTable, mplsTunnelARHopTable, and mplsTunnelCHopTable, respectively. By traversing the MPLS TE MIB in this manner using a hop table column and primary index, you can retrieve information pertaining to the hops of that tunnel configuration.

Because tunnels are treated as interfaces, the tunnel table column (mplsTunnelIfIndex) provides an index into the Interfaces MIB that you can use to retrieve interface-specific information about a tunnel.

How to Configure the MPLS Traffic Engineering MIB

Enabling the SNMP Agent to Help Manage Various MPLS TE Tunnel Characteristics of Tunnels on the Local Router

The SNMP agent for the MPLS TE MIB is disabled by default. To enable the SNMP agent for the MPLS TE MIB, perform the following task.

SUMMARY STEPS

1. **telnet** *host*
2. **enable**
3. **show running-config**
4. **configure terminal**
5. **snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [**ipv6** *nacl*] [*access-list-number*]
6. **snmp-server enable traps** [*identification-type*] [*notification-option*]
7. **exit**
8. **write memory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**telnet** *host* **Example:** `Router> telnet 192.172.172.172`	Telnets to the router identified by the specified IP address (represented as *xxx.xxx.xxx.xxx*).
Step 2	**enable** **Example:** `Router# enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 3	**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, go to Step 4 . If any SNMP information is displayed, you can modify the information or change it as needed.

	Command or Action	Purpose
Step 4	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 5	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [**ipv6** *nacl*] [*access-list-number*] **Example:** `Router(config)# snmp-server community comaccess ro 4`	Enables the read-only (RO) community string.
Step 6	**snmp-server enable traps** [*identification-type*] [*notification-option*] **Example:** `Router(config)# snmp-server enable traps`	Enables an LSR to send SNMP notifications or informs to an SNMP host. **Note** This command is optional. After SNMP is enabled, all MIBs (not just the TE MIB) are available for the user to query.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 8	**write memory** **Example:** `Router# write memory`	Writes the modified configuration to NVRAM, permanently saving the settings.

Verifying the Status of the SNMP Agent

To verify that the SNMP agent has been enabled on a host network device, perform the following steps.

SUMMARY STEPS

1. **telnet** *host*
2. **enable**
3. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**telnet** *host* **Example:** `Router# telnet 192.172.172.172`	Telnets to the target device identified by the specified IP address (represented as *xxx.xxx.xxx.xxx*).
Step 2	**enable** **Example:** `Router# enable`	Enables SNMP on the target device.
Step 3	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration on the target device and is used to examine the output for displayed SNMP information.

Examples

The follows example displays the running configuration on the target device and its SNMP information.

```
Router# show running-config
.
.
.
snmp-server community public ro
snmp-server community private ro
```

Any **snmp-server** statement that appears in the output and takes the form shown here verifies that SNMP has been enabled on that device.

Configuration Examples for the MPLS Traffic Engineering MIB

Example Enabling the SNMP Agent to Help Manage MPLS TE Characteristics of Tunnels on the Local Router

The following example shows how to enable an SNMP agent on a host network device:

```
Router# configure terminal
Router(config)# snmp-server community private
```

The following example shows how to enable SNMPv1 and SNMPv2C. The configuration permits any SNMP agent to access all MPLS TE MIB objects with read-only permissions using the community string public.

```
Router(config)# snmp-server community public
```

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

```
Router(config)# snmp-server community comaccess ro 4
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Information about MPLS TE and enhancements	MPLS Traffic Engineering and Enhancements
MPLS TE commands	*Multiprotocol Label Switching Command Reference*
SNMP commands	*Network Management Command Reference*
SNMP configuration	"Configuring SNMP Support" in the *Network Management Configuration Guide*

Standards

Standard	Title
draft-ietf-mpls-te-mib-05	*MPLS Traffic Engineering Management Information Base Using SMIv2*

MIBs

MIB	MIBs Link
• MPLS TE MIB • Interfaces 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 2026	*The Internet Standards Process*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for the MPLS Traffic Engineering MIB

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 58: Feature Information for the MPLS Traffic Engineering MIB

Feature Name	Releases	Feature Information
MPLS Traffic Engineering MIB	12.0(17)S 12.0(17)ST 12.2(8)T 12.2(14)S 12.2(28)SB 12.2(31)SB2	The MPLS Traffic Engineering MIB feature enables the SNMP agent support in Cisco IOS software for MPLS TE management, as implemented in the MPLS TE MIB. In 12.0(17)S, this feature provided the ability to generate and queue SNMP notification messages that signal changes in the operational status of MPLS TE tunnels when you are using the MPLS TE MIB on Cisco 7500 series routers and Cisco 12000 series Internet routers. In 12.0(17)ST, support for SNMP traffic engineering notifications was extended to include Cisco 7500 series routers and Cisco 12000 series Internet routers. In 12.2(8)T, support for SNMP TE notifications was extended to include Cisco 7500 series routers. The **snmp-server host** command was modified. In 12.2(14)S, this feature was integrated. In 12.2(28)SB, this feature was integrated. In 12.2(31)SB2, this feature was integrated. The following commands were introduced or modified: **snmp-server community**, **snmp-server enable traps mpls traffic-eng**, **snmp-server host**.

Glossary

affinity bits --A Multiprotocol Label Switching (MPLS) traffic engineering (TE) tunnel's requirements on the attributes of the links it will cross. The tunnel's affinity bits and affinity mask must match with the attributes of the various links carrying the tunnel.

call admission precendence --An Multiprotocol Label Switching (MPLS) traffic engineering tunnel with a higher priority will, if necessary, preempt an MPLS traffic engineering tunnel with a lower priority. An expected use is that tunnels that are more difficult to route will have a higher priority, and can preempt tunnels that are less difficult to route, on the assumption that those lower priority tunnels can find another path.

constraint-based routing--Procedures and protocols used to determine a route across a backbone taking into account resource requirements and resource availability, instead of simply using the shortest path.

flow --A traffic load entering the backbone at one point--point of presence (POP)--and leaving it from another that must be traffic engineered across the backbone. The traffic load will be carried across one or more LSP tunnels running from the entry POP to the exit POP.

headend --The label switch router (LSR) at which the tunnel originates. The tunnel's "head" or tunnel interface will reside at this LSR as well.

informs --A type of notification message that is more reliable than a conventional trap notification message because an informs message requires acknowledgment.

label --A short, fixed-length data construct that tells switching nodes how to forward data (packets or cells).

label switched path (LSP) tunnel--A configured connection between two routers, using label switching to carry the packets.

LSP --label switched path. A path that is followed by a labeled packet over several hops, starting at an ingress label switch router (LSR) and ending at an egress LSR.

LSR --label switch router. A Layer 3 router that forwards a packet based on the value of a label encapsulated in the packet.

MIB --Management Information Base. A database of network management information (consisting of MIB objects) that is used and maintained by a network management protocol such as Simple Network Management Protocol (SNMP). The value of a MIB object can be changed or retrieved using SNMP commands, usually by a GUI-based network management system. MIB objects are organized in a tree structure that includes public (standard) and private (proprietary) branches.

MPLS --Multiprotocol Label Switching. Switching method that forwards IP traffic using a label. This label instructs the routers and the switches in the network where to forward the packets based on preestablished IP routing information.

NMS --network management station. An NMS is a powerful, well-equipped computer (typically an engineering workstation) that is used by a network administrator to communicate with other devices in the network. An NMS is typically used to manage network resources, gather statistics, and perform a variety of network administration and configuration tasks.

notification --A message sent by a Simple Network Management Protocol (SNMP) agent to a network management station, console, or terminal to indicate that a significant event within Cisco software has occurred (see traps).

OSPF --Open Shortest Path First. A link-state routing protocol used for routing IP.

RSVP --Resource Reservation Protocol. Protocol for reserving network resources to provide quality of service (QoS) guarantees to application flows.

SNMP --Simple Network Management Protocol. A network management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, manage configurations, collect statistics, monitor performance, and ensure network security.

tailend --The downstream, receive end of a tunnel.

traffic engineering --Techniques and processes that cause routed traffic to travel through the network on a path other than the one that would have been chosen if standard routing methods were used.

trap --A message sent by a Simple Network Management Protocol (SNMP) agent to a network management station, console, or terminal to indicate that a significant event within Cisco software has occurred. Traps (notifications) are less reliable than inform requests, because the receiver of the trap does not send an acknowledgment of receipt; furthermore, the sender of the trap cannot determine if the trap was received (see notification).

VCC --virtual channel connection. A VCC is a logical circuit consisting of VCLs that carries data between two endpoints in an ATM network. Sometimes called a virtual circuit connection.

VCI --virtual channel identifier. A 16-bit field in the header of an ATM cell. The VCI, together with the VPI, is used to identify the next network VCL as the cell passes through a series of ATM switches on its way to its final destination.

VCL --virtual channel link. A VCL is the logical connection that exists between two adjacent switches in an ATM network.

VPI --virtual path identifier. An 8-bit field in the header of an ATM cell. The VPI, together with the VCI, is used to identify the next network VCL as the cell passes through a series of ATM switches on its way to its final destination.

CHAPTER 9

MPLS Traffic Engineering--Fast Reroute MIB

The MPLS Traffic Engineering--Fast Reroute MIB provides Simple Network Management Protocol (SNMP)-based network management of the Multiprotocol Label Switching (MPLS) Fast Reroute (FRR) feature in Cisco software.

The Fast Reroute MIB has the following features:

- Notifications can be created and queued.
- Command-line interface (CLI) commands enable notifications, and specify the IP address to where the notifications will be sent.
- The configuration of the notifications can be written into nonvolatile memory.

The MIB includes objects describing features within MPLS FRR, and it includes the following tables:

- cmplsFrrConstTable
- cmplsFrrLogTable
- cmplsFrrFacRouteDBTable

The MIB also includes scalar objects (that is, objects that are not in a table). For more information, see the FRR MIB Scalar Objects, on page 276.

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 the MPLS Traffic Engineering--Fast Reroute MIB

- The network must support the Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF) protocol.
- The SNMP is installed and enabled on the label switch routers (LSRs).
- MPLS is enabled globally on each LSR.
- Cisco Express Forwarding is enabled on the LSRs.
- Traffic engineering (TE) tunnels are enabled.
- MPLS FRR is enabled on one of the TE tunnels.
- The Resource Reservation Protocol (RSVP) is enabled.

Restrictions for the MPLS Traffic Engineering--Fast Reroute MIB

- The implementation of the FRR MIB is limited to read-only (RO) permission for MIB objects.
- Configuration of the FRR MIB using the SNMP SET command is not supported in Cisco IOS Release 12.2(33)SRA or in prior releases.
- The following tables are not implemented in the specified releases:
    - mplsFrrOne2OnePlrTable--Not implemented in Cisco IOS software.
    - mplsFrrDetourTable--Not implemented in Cisco IOS software.
    - cmplsFrrLogTable--Implemented only in Cisco IOS 12.0S-based releases.

Information About the MPLS Traffic Engineering--Fast Reroute MIB

Feature Design of the MPLS Traffic Engineering--Fast Reroute MIB

The FRR MIB enables standard, SNMP-based network management of FRR in Cisco software. This capability requires that SNMP agent code executes 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 FRR MIB is based on the Internet Engineering Task Force (IETF) draft MIB specification *draft-ietf-mpls-fastreroute-mib-02.txt* . The IETF draft MIB, which undergoes revisions periodically, is evolving toward becoming a standard. The Cisco implementation of the FRR MIB is expected to track the evolution of the IETF draft MIB, and may change accordingly.

Slight differences between the IETF draft MIB and the implementation of FRR within Cisco software require some minor translations between the FRR MIB objects and the internal data structures of Cisco software. These translations are accomplished by the SNMP agent, which runs in the background on the NMS workstation as a low priority process and provides a management interface to Cisco software.

You can use an SNMP agent to access FRR MIB objects using standard SNMP GET operations. All the objects in the FRR MIB follow the conventions defined in the IETF draft MIB.

Functional Structure of the MPLS Traffic Engineering--Fast Reroute MIB

The SNMP agent code supporting the FRR MIB follows the existing model for such code in Cisco software and is, in part, generated by the Cisco tool set, based on the MIB source code. The basis for the generated code is the Cisco version of the FRR MIB CISCO-ietf-frr-mib.

The SNMP agent code, which has a layered structure that is common to MIB support code in Cisco software, consists of the following layers:

- Platform-independent layer--This layer is generated primarily by the MIB development Cisco tool set and incorporates platform- and implementation-independent functions. These functions handle SNMP standard functionality in the context of the specific MIB. This layer handles indexes and range or enumeration value checks for GET, GET-NEXT, and SET SNMP operations. A function is generated for each SNMP table or group of objects. This layer calls into the next layer.
- Application interface layer--The Cisco tool set generates the function names and template code for MIB objects.
- Application-specific layer--This layer provides the mechanism for retrieving relevant data from the managed application layer. It includes an entry point function for each table. This function calls two other functions; one that searches the TE tunnel database that RSVP maintains for the relevant data according to the indexes, and another function that fills the data into the structure.
- Managed application layer--This layer includes all the structures and mechanisms, and is managed by the MIB.

System Flow of SNMP Protocol Requests and Response Messages

All SNMP protocol requests and response messages are ultimately handled by the SNMP master agent. When such a message is received on a router, the master agent parses the requests and identifies the MIB to which the request refers. The master agent then queries the subagent responsible for the MIB with a GET, GET-NEXT, or SET request. The FRR MIB subagent retrieves the appropriate data, and returns it to the master agent. The master agent is then responsible for returning an SNMP response to the NMS. All queries occur within the IP SNMP Cisco software process, which runs as a low priority task.

FRR MIB Scalar Objects

Scalar objects are objects that are not in tables. A scalar object has one instance (that is, one occurrence).

The table below describes the FRR MIB scalar objects.

Table 59: Scalar Objects

MIB Object	Function
cmplsFrrDetourIncoming	Number of detour link-state packets (LSPs) entering the device. This object returns 0 because cmplsFrrConstProtectionMethod is set to facilityBackup(1).
cmplsFrrDetourOutgoing	Number of detour LSPs leaving the device. This object returns 0 because cmplsFrrConstProtectionMethod is set to facilityBackup(1).
cmplsFrrDetourOriginating	Number of detour LSPs originating from the device. This object returns 0 because cmplsFrrConstProtectionMethod is set to facilityBackup(1).
cmplsFrrSwitchover	Number of tunnels that are being backed up because cmplsFrrConstProtectionMethod is set to facilityBackup(1).
cmplsFrrNumOfConfIfs	Number of MPLS interfaces FRR configured for protection; 0 indicates that LSPs traversing any interface can be protected.
cmplsFrrActProtectedIfs	Number of interfaces FRR is protecting because cmplsFrrConstProtectionMethod is set to facilityBackup(1).
cmplsFrrConfProtectingTuns	Number of backup Fast Reroute-protected tunnels configured because cmplsFrrConstProtectionMethod is set to facilityBackup(1).
cmplsFrrActProtectedTuns	Number of tunnels protected by the Fast Reroute feature. This object returns 0 because cmplsFrrConstProtectionMethod is set to facilityBackup(1).

MIB Object	Function
cmplsFrrActProtectedLSPs	Number of LSPs that FRR is protecting. If cmplsFrrConstProtectionMethod is set to facilityBackup(1), this object returns 0.
cmplsFrrConstProtectionMethod	This object always returns facilityBackup(1) because Cisco software supports only the facility backup protection method.
cmplsFrrNotifsEnabled	A value that indicates whether FRR notifications defined in this MIB are enabled or disabled. This object returns True(1) for enabled, or False(2) for disabled. The default is that notifications are disabled.
cmplsFrrLogTableMaxEntries	Maximum number of entries allowed in the FRR log table.
cmplsFrrLogTableCurrEntries	Current number of entries in the FRR log table. This object always returns 0.
cmplsFrrNotifMaxRate	Maximum interval rate between FRR MIB notifications. This object always returns 0.

FRR MIB Notification Generation Events

Notifications are issued after particular FRR events occur.

When you enable FRR MIB notification functionality by issuing the **snmp-server enable traps mpls fast-reroute** command, FRR events generate notification messages that are sent to a designated NMS in the network to signal the occurrence of specific events in Cisco software.

The FRR MIB objects involved in FRR status transitions and event notifications include cmplsFrrProtected. This message is sent to an NMS if there is a major TE tunnel change (that is, fast rerouting of TE tunnels).

FRR MIB Notification Specification

Notifications are issued after particular FRR events occur.

Each FRR notification has a generic type identifier and an enterprise-specific type identifier for identifying the notification type. The generic type for all FRR notifications is “enterprise Specific” because this is not one of the generic notification types defined for SNMP. The enterprise-specific type is 1 for cmplsFrrProtected.

Each notification contains the following objects from the FRR MIB so that the FRR tunnel can be easily identified:

- cmplsFrrConstNumProtectingTunOnIf
- cmplsFrrConstNumProtectedTunOnIf
- cmplsFrrConstBandwidth

Upon being invoked, the appropriate FRR interface indexes have already been retrieved by existing FRR code. The FRR interfaces are then used to fill in data for the three objects included in the notification.

FRR MIB Notification Monitoring

Notifications are issued after particular FRR events occur.

When FRR MIB notifications are enabled (see the **snmp-server enable traps** command), notification messages relating to specific FRR events within Cisco software are generated and sent to a specified NMS in the network. Any utility that supports SNMPv1 or SNPv2 notifications can receive notification messages.

To monitor FRR MIB notifications, log in to an NMS that supports a utility that displays SNMP notifications, and start the display utility.

MIB Tables in the MPLS Traffic Engineering--Fast Reroute MIB

The FRR MIB consists of the following tables:

The tables access various data structures to obtain information regarding detours, the FRR database, and logging.

cmplsFrrConstTable

cmplsFrrConstTable displays the configuration of an FRR-enabled tunnel and the characteristics of its accompanying backup tunnels. For each protected tunnel, there can be multiple backup tunnels.

The table is indexed by the following:

- cmplsFrrConstIfIndex
- cmplsFrrConstTunnelIndex
- cmplsFrrConstTunnelInstance

The table below describes the MIB objects for cmplsFrrConstTable.

Table 60: cmplsFrrConstTable Objects

MIB Object	Function
cmplsFrrConstIfIndex	Uniquely identifies an interface on which FRR is configured. If an index has a value of 0, the configuration applies to all interfaces on the device on which the FRR feature can operate.
cmplsFrrConstTunnelIndex	Tunnel for which FRR is requested.
cmplsFrrConstTunnelInstance	Tunnel for which FRR is requested. The value always is 0 because only tunnel heads are represented, and tunnel heads have an instance value of 0.
cmplsFrrConstSetupPrio	Setup priority of the backup tunnel.

MIB Object	Function
cmplsFrrConstHoldingPrio	Holding priority of the backup tunnel.
cmplsFrrConstInclAnyAffinity	Attribute bits that must be set for the tunnel to traverse a link.
cmplsFrrConstInclAllAffinity	Attribute bits that must not be set for the tunnel to traverse a link.
cmplsFrrConstExclAllAffinity	A link satisfies the exclude-all constraint only if the link contains none of the administrative groups specified in the constraint.
cmplsFrrConstHopLimit	The maximum number of hops that the backup tunnel can traverse.
cmplsFrrConstBandwidth	The bandwidth of the backup tunnels for this tunnel, in thousands of bits per second (kbps).
cmplsFrrConstRowStatus	Creates, modifies, and deletes a row in this table.

cmplsFrrLogTable

Note cmplsFrrLogTable and the **show mpls traffic-eng fast-reroute log reroutes**command are supported only in Cisco IOS 12.0S-based releases.

cmplsFrrLogTable is indexed by the object cmplsFrrLogIndex. The index corresponds to a log entry in the FRR feature's **show mpls traffic-eng fast-reroute log reroutes** command. That **show** command stores up to 32 entries at a time. If entries are added, the oldest entry is overwritten with new log information.

cmplsFrrLogTable can store up to 32 entries at a time, overwriting older entries as newer ones are added. The index cmplsFrrLogIndex is incremented to give each log table entry of the MIB a unique index value. Therefore, it is possible to have indexes greater than 32 even though only 32 entries are displaying.

The table below describes the MIB objects for cmplsFrrLogTable.

Table 61: cmplsFrrLogTable Objects

MIB Object	Function
cmplsFrrLogIndex	Number of the FRR event.
cmplsFrrLogEventTime	Number of milliseconds that elapsed from bootstrap time to the time that the event occurred.
cmplsFrrLogInterface	Identifies the interface that was affected by this FRR event. The value can be set to 0 if mplsFrrConstProtectionMethod is set to oneToOneBackup(0).
cmplsFrrLogEventType	The type of FRR event that occurred. The object returns Protected or Other.

MIB Object	Function
cmplsFrrLogEventDuration	Duration of the event, in milliseconds.
cmplsFrrLogEventReasonString	Implementation-specific explanation of the event. The object returns interface down event or interface other event.

cmplsFrrFacRouteDBTable

The following indexes specify which interface and tunnel are being protected by the FRR feature:

- cmplsFrrFacRouteProtectedIfIndex
- cmplsFrrFacRouteProtectedTunIndex

The following indexes specify the backup tunnel that provides protection to the protected tunnel:

- cmplsFrrFacRouteProtectedIfIndex
- cmplsFrrFacRouteProtectingTunIndex
- cmplsFrrFacRouteProtectedTunIndex
- cmplsFrrFacRouteProtectedTunInstance
- cmplsFrrFacRouteProtectedTunIngressLSRId
- cmplsFrrFacRouteProtectedTunEgressLSRId

This version of the MIB will attempt to leverage the work already done for the MPLS TE MIB because it contains similar lookup functions for TE tunnels.

The table below describes the MIB objects for cmplsFrrFacRouteDBTable.

Table 62: cmplsFrrFacRouteDBTable Objects

MIB Object	Function
cmplsFrrFacRouteProtectedIfIndex	Interface configured for FRR protection.
cmplsFrrFacRouteProtectingTunIndex	The tunnel number of the protecting (backup) tunnel.
cmplsFrrFacRouteProtectedTunIndex	The mplsTunnelEntry primary index for the tunnel head interface designated to protect the interface specified in mplsFrrFacRouteIfProtIdx (and all the tunnels using this interface).
cmplsFrrFacRouteProtectedTunInstance	An mplsTunnelEntry that is being protected by FRR. An instance uniquely identifies a tunnel.
cmplsFrrFacRouteProtectedTunIngressLSRId	Inbound label for the backup LSR.
cmplsFrrFacRouteProtectedTunEgressLSRId	Outbound label for the backup LSR.

MIB Object	Function
cmplsFrrFacRouteProtectedTunStatus	State of the protected tunnel. Valid values are: • active--Tunnel label has been placed in the Label Forwarding Information Base (LFIB) and is ready to be applied to incoming packets. • ready--Tunnel's label entry has been created, but is not in the LFIB. • partial--Tunnel's label entry has not been fully created.
cmplsFrrFacRouteProtectingTunResvBw	Amount of bandwidth, in megabytes per second, that is reserved by the backup tunnel.
cmplsFrrFacRouteProtectingTunProtectionType	Type of protection: 0 designates link protection; 1 designates node protection.

How to Configure the MPLS Traffic Engineering--Fast Reroute MIB

Enabling the SNMP Agent for FRR MIB Notifications

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **configure terminal**
4. **snmp-server community** *string* [**view** *view-name*] [**ro**] [*access-list-number*]
5. **snmp-server enable traps mpls fast-reroute protected**
6. **end**
7. **write memory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration of the router to determine if an SNMP agent is already running on the device. If no SNMP information is displayed, continue with the next step. If any SNMP information is displayed, you can modify or change the information.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server community** *string* [**view** *view-name*] [**ro**] [*access-list-number*] **Example:** `Router(config)# snmp-server community public ro`	Configures read-only (ro) SNMP community strings for the FRR MIB.
Step 5	**snmp-server enable traps mpls fast-reroute protected** **Example:** `Router(config)# snmp-server enable traps mpls fast-reroute protected`	Enables Fast Reroute traps.

Step 6	**end**
	
	Example:
	
	`Router(config)# end`
		Exits to privileged EXEC mode.

Step 7	**write memory** **Example:** `Router# write memory`	Writes the modified SNMP configuration into NVRAM of the router, permanently saving the SNMP settings.

Enabling Cisco Express Forwarding

SUMMARY STEPS

1. **enable**
2. **configure teminal**
3. **ip cef distributed**
4. **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 teminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip cef distributed** **Example:** `Router(config)# ip cef distributed`	Enables distributed Cisco Express Forwarding.

Step 4	**end**
	
	Example:
	
	`Router(config)# end`
		Exits to privileged EXEC mode.

Enabling TE Tunnels

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip cef**
4. **mpls traffic-eng tunnels**
5. **interface** *typeslot/port*
6. **mpls traffic-eng tunnels**
7. **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	**ip cef** **Example:** `Router(config)# ip cef`	Enables standard Cisco Express Forwarding operations.
Step 4	**mpls traffic-eng tunnels** **Example:** `Router(config)# mpls traffic-eng tunnels`	Enables the MPLS TE tunnel feature on a device.
Step 5	**interface** *typeslot/port* **Example:** `Router(config)# interface POS1/0`	Specifies the interface and enters interface configuration mode.

	Command or Action	Purpose
Step 6	**mpls traffic-eng tunnels** **Example:** `Router(config-if)# mpls traffic-eng tunnels`	Enables the MPLS TE tunnel feature on an interface.

Step 7	**end**
	
	Example:
	
	`Router(config-if)# end`
		Returns to privileged EXEC mode.

Enabling MPLS FRR on Each TE Tunnel

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typeslot/port*
4. **tunnel mode mpls traffic-eng**
5. **tunnel mpls traffic-eng fast-reroute**
6. **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	**interface** *typeslot/port* **Example:** `Router(config)# interface POS1/0`	Specifies the interface and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**tunnel mode mpls traffic-eng** **Example:** `Router(config-if)# tunnel mode mpls traffic-eng`	Sets the mode of a tunnel to MPLS for traffic engineering.
Step 5	**tunnel mpls traffic-eng fast-reroute** **Example:** `Router(config-if)# tunnel mpls traffic-eng fast-reroute`	Enables Fast Reroute on the TE tunnel being protected.

Step 6	**end**
	
	Example:
	
	`Router(config-if)# end`
		Exits to privileged EXEC mode.

Enabling a Backup Tunnel on an Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typeslot/port*

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.

	Command or Action	Purpose
Step 3	**interface** *typeslot/port* **Example:** `Router(config)# interface POS1/0`	Specifies the interface and enters interface configuration mode.

Configuration Examples for the MPLS Traffic Engineering--Fast Reroute MIB

Enabling an SNMP Agent on a Host NMS Example

The following example shows how to enable an SNMP agent on the host NMS:

```
enable
 show running-config
 configure terminal
 snmp-server community public ro
 snmp-server enable traps mpls fast-reroute protected
 end
 write memory
```

Enabling Cisco Express Forwarding Example

The following example shows how to enable Cisco Express Forwarding:

```
enable
 configure terminal
 ip cef distributed
 end
```

Enabling TE Tunnels Example

The following example shows how to enable traffic engineering tunnels:

```
enable
 configure terminal
 ip cef
 mpls traffic-eng tunnels
 interface Ethernet1/0
 mpls traffic-eng tunnels
 end
```

Enabling MPLS FRR on Each TE Tunnel Example

The following example shows how to enable MPLS Fast Reroute on each TE tunnel:

```
enable
 configure terminal
 interface POS1/0
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng fast-reroute
 end
```

Enabling a Backup Tunnel on an Interface Example

The following example shows how to enable a backup tunnel on an interface:

```
enable
 configure terminal
 interface POS1/0
 mpls traffic-eng backup-path tunnel1
 end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Description of commands associated with MPLS and MPLS applications	*Multiprotocol Label Switching Command Reference*
SNMP agent support for the MPLS Traffic Engineering MIB	MPLS Traffic Engineering MIB
Fast Reroute	MPLS Traffic Engineering: Fast Reroute Link and Node Protection

Standards

Standard	Title
MPLS-FRR-MIB	*draft-ietf-mpls-fastreroute-mib-02.txt*

MIBs

MIB	MIBs Link
MPLS Traffic Engineering (TE) 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
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for MPLS Traffic Engineering--Fast Reroute MIB

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 63: Feature Information for MPLS Traffic Engineering--Fast Reroute MIB

Feature Name	Releases	Feature Information
MPLS Traffic Engineering--Fast Reroute MIB	12.0(10)ST 12.0(16)ST 12.0(22)S 12.0(26)S 12.2(33)SRA 12.2(33)SXH 12.4(20)T 15.3(2)S	The MPLS Traffic Engineering--Fast Reroute MIB provides SNMP-based network management of the Multiprotocol Label Switching (MPLS) Fast Reroute (FRR) feature in Cisco IOS software. In 12.0(10)ST, the Fast Reroute link protection feature was introduced. In 12.0(16)ST, link protection for Cisco series 7200 and 7500 platforms was added. In 12.0(22)S, Fast Reroute enhancements, including node protection, were added. In 12.0(26)S, support for the IETF MIB *draft-ietf-mpls-fastreroute-mib-02.txt,* which provides network management for the FRR feature, was added. In 12.2(33)SRA, support for cmplsFrrLogTableCurrEntries and cmplsFrrNotifMaxRate was added. The cmplsFrrLogTable is not supported. In 12.2(33)SXH, support was added. In 12.4(20)T, support was added. In 15.3(2)S, this feature was integrated.

Glossary

Cisco Express Forwarding --An advanced Layer 3 IP switching technology. Cisco Express Forwarding optimizes network performance and scalability for networks with large and dynamic traffic patterns.

index --A method of uniquely identifying a tunnel.

instance --An occurrence. An object can have one or more instances.

IS-IS --Intermediate System-to-Intermediate System. IS-IS is an OSI link-state hierarchical routing protocol based on DECnet Phase V routing where intermediate system (IS) routers exchange routing information based on a single metric to determine network topology.

label --A short, fixed-length data construct that tells switching nodes how to forward data (packets or cells).

LFIB --Label Forwarding Information Base. The data structure for storing information about incoming and outgoing tags (labels) and associated equivalent packets suitable for labeling.

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

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

NMS --network management station. A powerful, well-equipped computer (typically an engineering workstation) that is used by a network administrator to communicate with other devices in the network. An NMS is typically used to manage network resources, gather statistics, and perform a variety of network administration and configuration tasks.

notification --A message sent by a Simple Network Management Protocol (SNMP) agent to a network management station, console, or terminal to indicate that a significant event within Cisco software has occurred.

object --A variable that has a specific instance associated with it.

OSPF --Open Shortest Path First. Link-state, hierarchical Interior Gateway Protocol (IGP) routing algorithm proposed as a successor to Routing Information Protocol (RIP) in the Internet community. OSPF features include least-cost routing, multipath routing, and load balancing.

RSVP --Resource Reservation Protocol. Protocol for reserving network resources to provide quality of service (QoS) guarantees to application flows.

scalar object--Objects that are not instances. A scalar object has one instance.

SNMP --Simple Network Management Protocol. A network management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, manage configurations, collect statistics, monitor performance, and ensure network security.

SNMP agent--A managed node or device. The router that has the MIB implementation on it.

CHAPTER 10

MPLS VPN--MIB Support

This document describes the Simple Network Management Protocol (SNMP) agent support in Cisco software for Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) management, as implemented in the draft *MPLS/BGP Virtual Private Network Management Information Base Using SMIv2* (*draft-ietf-ppvpn-mpls-vpn-mib-05.txt*). This document also describes the cMplsNumVrfRouteMaxThreshCleared notification, which is implemented as part of the proprietary MIB CISCO-IETF-PPVNP-MPLS-VPN-MIB.

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 VPN--MIB Support

- SNMP is installed and enabled on the label switching routers.
- MPLS is enabled on the label switching routers.
- Multiprotocol Border Gateway Protocol (BGP) is enabled on the label switching routers.
- Cisco Express Forwarding is enabled on the label switching routers.

Restrictions for MPLS VPN--MIB Support

- Configuration of the MIB using the **snmp set**command is not supported, except for trap-related objects, such as mplsVpnNotificationEnable and mplsVpnVrfSecIllegalLabelRcvThresh.
- The mplsVpnVrfBgpNbrPrefixTable is not supported.

Information About MPLS VPN--MIB Support

MPLS VPN Overview

The MPLS VPN technology allows service providers to offer intranet and extranet VPN services that directly connect their customers’ remote offices to a public network with the same security and service levels that a private network offers. Each VPN is associated with one or more VPN routing and forwarding (VRF) instances. A VRF is created for each VPN defined on a router and contains most of the information needed to manage and monitor MPLS VPNs: an IP routing table, a derived Cisco Express Forwarding table, a set of interfaces that use this forwarding table, and a set of rules and routing protocol parameters that control the information that is included in the routing table.

MPLS VPN MIB Overview

The Provider-Provisioned VPN (PPVPN)-MPLS-VPN MIB provides access to MPLS VRF information, and interfaces included in the VRF, and other configuration and monitoring information.

The PPVPN-MPLS-VPN MIB provides the following benefits:

- A standards-based SNMP interface for retrieving information about critical MPLS VPN events.
- VRF information to assist in the management and monitoring of MPLS VPNs.
- Information, in conjunction with the Interfaces MIB, about interfaces assigned to VRFs.
- Performance statistics for all VRFs on a router.
- The generation and queueing of notifications that call attention to major changes in the operational status of MPLS VPN enabled interfaces; the forwarding of notification messages to a designated network management system (NMS) for evaluation and action by network administrators.

- Advanced warning when VPN routing tables are approaching or exceed their capacity.
- Warnings about the reception of illegal labels on a VRF-enabled interface. Such receptions may indicate misconfiguration or an attempt to violate security.

This document also describes the CISCO-IETF-PPVPN-MPLS-VPN-MIB, which contains the cMplsNumVrfRouteMaxThreshCleared notification.

MPLS VPN MIB and the IETF

SNMP agent code operating with the PPVPN-MPLS-VPN MIB enables a standardized, SNMP-based approach to managing MPLS VPNs in Cisco software.

The PPVPN-MPLS-VPN MIB is based on the Internet Engineering Task Force draft MIB specification *draft-ietf-ppvpn-mpls-vpn-mib-05.txt* , which includes objects describing features that support MPLS VPN events. This IETF draft MIB, which undergoes revisions from time to time, is becoming a standard. Accordingly, the Cisco implementation of the PPVPN-MPLS-VPN MIB is expected to track the evolution of the IETF draft MIB, and may change accordingly.

Some slight differences between the IETF draft MIB and the actual implementation of MPLS VPNs within Cisco software require some minor translations between the PPVPN-MPLS-VPN MIB and the internal data structures of Cisco software. These translations are accomplished by means of the SNMP agent code. Also, while running as a low priority process, the SNMP agent provides a management interface to Cisco software. SNMP adds little overhead on the normal functions of the device.

The SNMP objects defined in the PPVPN-MPLS-VPN MIB can be viewed by any standard SNMP utility. The network administrator can retrieve information in the PPVPN-MPLS-VPN MIB using standard SNMP get and getnext operations for SNMP v1, v2, and v3.

All PPVPN-MPLS-VPN MIB objects are based on the IETF draft MIB; thus, no Cisco-specific SNMP application is required to support the functions and operations pertaining to the PPVPN-MPLS-VPN MIB features.

Capabilities Supported by PPVPN-MPLS-VPN MIB

The PPVPN-MPLS-VPN MIB provides you with the ability to do the following:

- Gather routing and forwarding information for MPLS VPNs on a router.
- Expose information in the VRF routing table.
- Gather information on BGP configuration related to VPNs and VRF interfaces and statistics.
- Emit notification messages that signal changes when critical MPLS VPN events occur.
- Enable, disable, and configure notification messages for MPLS VPN events by using extensions to existing SNMP command-line interface (CLI) commands.
- Specify the IP address of NMS in the operating environment to which notification messages are sent.
- Write notification configurations into nonvolatile memory.

Functional Structure of the PPVPN-MPLS-VPN MIB

The SNMP agent code supporting the PPVPN-MPLS-VPN MIB follows the existing model for such code in Cisco software and is, in part, generated by the Cisco software tool set, based on the MIB source code.

The SNMP agent code, which has a layered structure that is common to MIB support code in Cisco software, consists of four layers:

- Platform-independent layer--This layer is generated primarily by the MIB development Cisco software tool set and incorporates platform- and implementation-independent functions. The Cisco MIB development tool set creates a standard set of files associated with a MIB.
- Application interface layer--The functions, names, and template code for MIB objects in this layer are also generated by the MIB development Cisco software tool set.
- Application-specific layer--This layer provides an interface between the application interface layer and the API and data structures layer below and performs tasks needed to retrieve required information from Cisco software, such as searching through data structures.
- API and data structures layer--This layer contains the data structures or APIs within Cisco software that are retrieved or called in order to set or retrieve SNMP management information.

Supported Objects in PPVPN-MPLS-VPN MIB

The PPVPN-MPLS-VPN MIB contains numerous tables and object definitions that provide read-only SNMP management support for the MPLS VPN feature in Cisco IOS software. The PPVPN-MPLS-VPN MIB conforms to Abstract Syntax Notation One (ASN.1), thus reflecting an idealized MPLS VPN database.

Using any standard SNMP network management application, you can retrieve and display information from the PPVPN-MPLS-VPN MIB using GET operations; similarly, you can traverse information in the MIB database for display using GETNEXT operations.

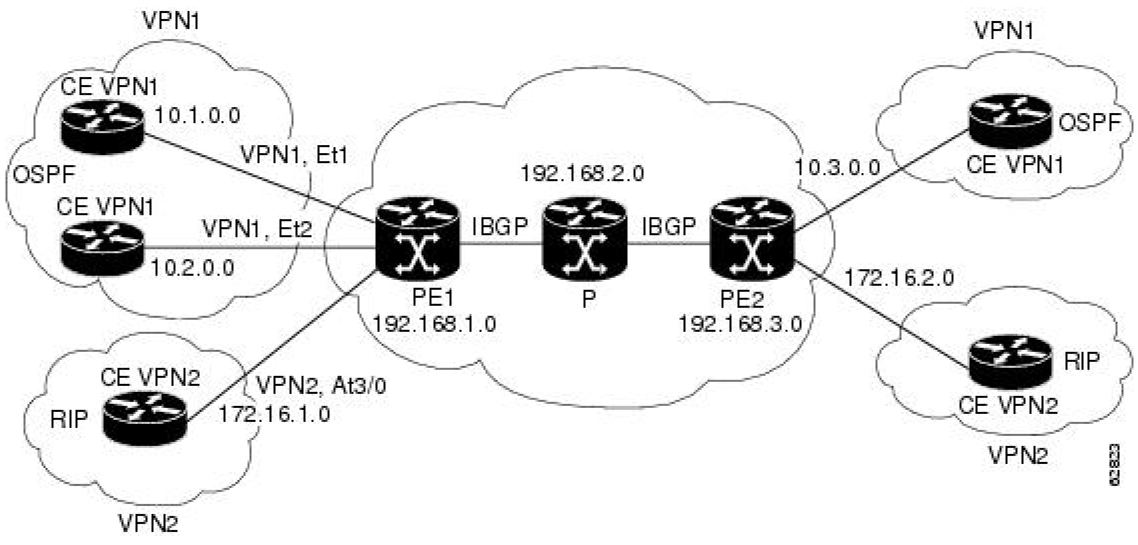
The PPVPN-MPLS-VPN MIB tables and objects are described briefly in the following sections:

The figure below shows a simple MPLS VPN configuration. This configuration includes two customer MPLS VPNs, labeled VPN1 and VPN2, and a simple provider network that consists of two provider edge (PE) routers, labeled PE1 and PE2, and a provider core router labeled P. The figure below shows the following sample configuration:

- VRF names--VPN1 and VPN2
- Interfaces associated with VRFs--Et1, Et2, and At3/0
- Routing protocols--Open Shortest Path First. Link-state (OSPF), Routing Information Protocol (RIP), and internal Border Gateway Protocol (IBGP)
- Routes associated with VPN1--10.1.0.0, 10.2.0.0, and 10.3.0.0
- Routes associated with VPN2--172.16.1.0 and 172.16.2.0
- Routes associated with the provider network--192.168.1.0, 192.168.2.0, and 192.168.3.0

This configuration is used in this document to explain MPLS VPN events that are monitored and managed by the PPVPN-MPLS-VPN MIB.

Figure 32: Sample MPLS VPN Configuration



Scalar Objects

The table below shows the supported PPVPN-MPLS-VPN MIB scalar objects.

Table 64: PPVPN-MPLS-VPN MIB Scalar Objects

MIB Object	Function
mplsVpnConfiguredVrfs	The number of VRFs configured on the router, including VRFs recently deleted.
mplsVpnActiveVrfs	The number of VRFs that are active on the router. An active VRF is assigned to at least one interface that is in the operationally up state.
mplsVpnConnectedInterfaces	The total number of interfaces assigned to any VRF.
mplsVpnNotificationEnable	A value that indicates whether all the PPVPN-MPLS-VPN MIB notifications are enabled: • Setting this object to true enables all notifications defined in the PPVPN-MPLS-VPN MIB. • Setting this object to false disables all notifications defined in the MIB. This is one of the few objects that is writable.

MIB Object	Function
mplsVpnVrfConfMaxPossibleRoutes	A number that indicates the amount of routes that this router is capable of storing. This value cannot be determined because it is based on the amount of available memory in the system. Therefore, this object is set to zero (0).

MIB Tables

The PPVPN-MPLS-VPN MIB implementation supports the following tables described in this section:

mplsVpnVrfTable

Each VRF is referenced by its VRF name (mplsVpnVrfName). The table below lists the MIB objects and their functions for this table.

Table 65: PPVPN-MPLS-VPN MIB Objects for the mplsVpnVrfTable

MIB Object	Function
mplsVpnVrfName	The name associated with this VRF. When this object is used as an index to a table, the first octet is the string length, and subsequent octets are the ASCII codes of each character. For example, “vpn1” is represented as 4.118.112.110.49.

mplsVpnVrfDescription	The description of the VRF. This is specified with the following configuration command:
	`Router(config)# ip vrf vrf-name`
	`Router(config-vrf)# description vrf-description`

mplsVpnVrfRouteDistinguisher	The route distinguisher for this VRF. This is specified with the following configuration command:
	`Router(config)# ip vrf vrf-name`
	`Router(config-vrf)# rd route-distinguisher`

mplsVpnVrfCreationTime	The value of the sysUpTime when this VRF entry was created.
mplsVpnVrfOperStatus	The operational status of this VRF. A VRF is up (1) when at least one interface associated with the VRF is up. A VRF is down (2) when: • No interfaces exist whose ifOperStatus = up (1). • No interfaces are associated with this VRF.

MIB Object	Function
mplsVpnVrfActiveInterfaces	The number of interfaces assigned to this VRF that are operationally up.
mplsVpnVrfAssociatedInterfaces	The number of interfaces assigned to this VRF, independent of the operational status.

mplsVpnVrfConfMidRouteThreshold	The middle route threshold. If the amount of routes in the VRF crosses this threshold, an mplsNumVrfRouteMidThreshExceeded notification is sent (if notifications are enabled and configured). You can set this value in configuration mode as a percentage of the maximum with the **maximum routes** *limit* {*warn-threshold* | **warn-only**} command, as follows:
	`Router(config)# ip vrf vpn1`
	`Router(config-vrf)# maximum routes 1000 50`
	The middle or warn threshold is set for VRF vpn1 as 50 percent of the maximum route threshold.
	The following command sets a middle threshold of 1000 routes. An mplsNumVrfRouteMidThreshExceeded notification is sent when this threshold is exceeded. However, additional routes are still allowed because a maximum route threshold is not set with this command.
	`Router(config-vrf)# maximum routes 1000 warn-only`

mplsVpnVrfConfHighRouteThreshold	The maximum route threshold. If the number of routes in the VRF crosses this threshold, an mplsNumVrfRouteMaxThreshExceeded notification is sent (if notifications are enabled and configured). You can set this value in configuration mode with the **maximum routes** *limit* {*warn-threshold* | **warn-only**} command as follows:
	`Router(config)# ip vrf vpn2`
	`Router(config-vrf)# maximum routes 1000 75`
	The maximum route threshold is set for 1000 routes for VRF vpn2 with a middle or warn threshold of 75 percent of this threshold.

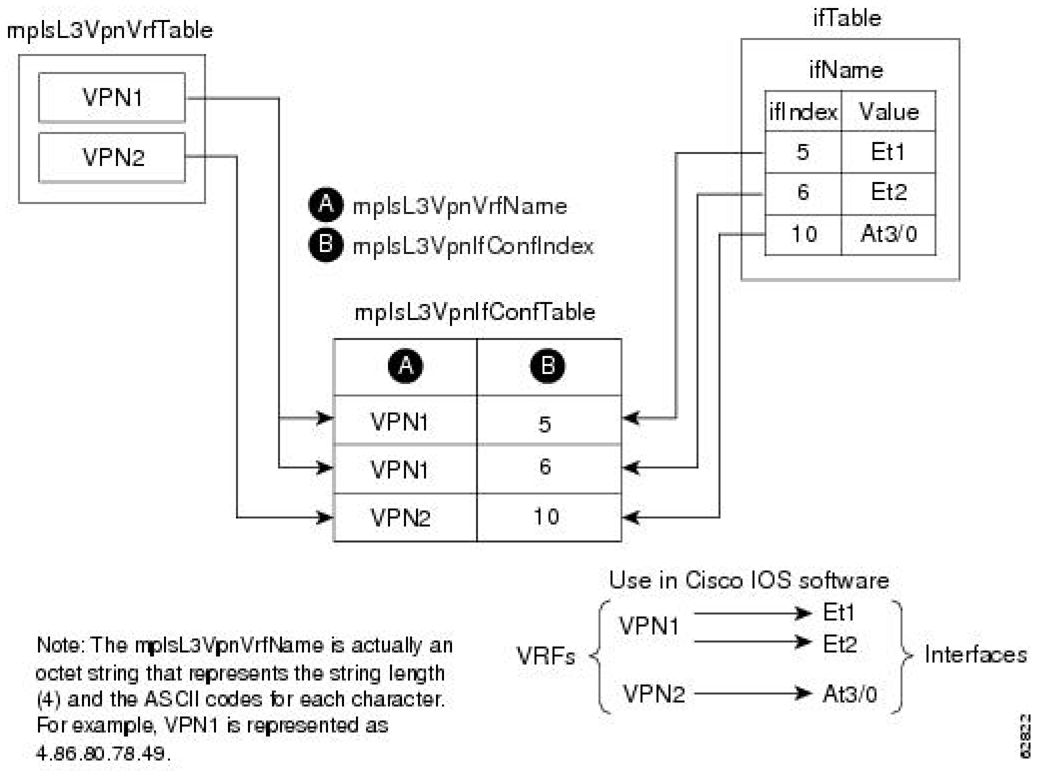
mplsVpnVrfConfMaxRoutes	This value is the same as the mplsVpnVrfConfHighRouteThreshold.
mplsVpnVrfConfLastChanged	The value of sysUpTime when the configuration of the VRF changes or interfaces are assigned or unassigned from the VRF. **Note** This object is updated only when values in this table change.
mplsVpnVrfConfRowStatus	Read-only implementation. This object normally reads “active (1),” but may read “notInService (2),” if a VRF was recently deleted.

MIB Object	Function
mplsVpnVrfConfStorageType	Read-only implementation. This object always reads "volatile (2)."

mplsVpnInterfaceConfTable

A VRF is associated with one MPLS VPN. Zero or more interfaces can be associated with a VRF. A VRF uses an interface that is defined in the ifTable of the Interfaces Group of MIB II (IFMIB). The IFMIB defines objects for managing interfaces. The ifTable of this MIB contains information on each interface in the network. The mplsVpnInterfaceConfTable associates a VRF from the mplsVpnVrfTable with a forwarding interface from the ifTable. The figure below shows the relationship between VRFs and interfaces defined in the ifTable and the mplsVpnInterfaceConfTable.

Figure 33: VRFs, the Interfaces MIB, and the mplsVpnInterfaceConfTable


mplsL3VpnVrfTable
VPN1
VPN2
ifTable
ifName
ifIndex Value
5 Et1
6 Et2
10 At3/0
A mplsL3VpnVrfName
B mplsL3VpnIfConfIndex
mplsL3VpnIfConfTable
A B
VPN1 5
VPN1 6
VPN2 10
Use in Cisco IOS software
VRFs
VPN1
VPN2
Et1
Et2
At3/0
Interfaces
Note: The mplsL3VpnVrfName is actually an octet string that represents the string length (4) and the ASCII codes for each character. For example, VPN1 is represented as 4.86.80.78.49.
62822

Entries in the VPN interface configuration table (mplsVpnInterfaceConfTable) represent the interfaces that are assigned to each VRF. The information available in this table is also displayed with the **show ip vrf** command.

The mplsVpnInterfaceConfTable shows how interfaces are assigned to VRFs. A label switch router (LSR) creates an entry in this table for every interface capable of supporting MPLS VPNs.

The mplsVpnInterfaceConfTable is indexed by the following:

- mplsVpnVrfName--The VRF name

- mplsVpnInterfaceConfIndex--An identifier that is the same as the ifIndex from the Interface MIB of the interface assigned to the VRF

The table below lists the MIB objects and their functions for this table.

Table 66: PPVPN-MPLS-VPN MIB Object for the mplsVpnInterfaceConfTable

MIB Object	Function
mplsVpnInterfaceConfIndex	Provides the interface MIB ifIndex of this interface that is assigned to a VRF.
mplsVpnInterfaceLabelEdgeType	Indicates whether the interface is a provider edge interface (1) or a customer edge interface (2). This value is always providerEdge (1) because in Cisco software, customerEdge interfaces are not assigned to VRFs and do not appear in this table.
mplsVpnInterfaceVpnClassification	Specifies what type of VPN this interface is providing: carrier supporting carrier (CSC) (1), enterprise (2), or InterProvider (3). This value is set to enterprise (2) if MPLS is not enabled and to carrier supporting carrier (1) if MPLS is enabled on this interface.
mplsVpnInterfaceVpnRouteDistProtocol	Indicates the route distribution protocols that are being used to redistribute routes with BGP on this interface: BGP (2), OSPF (3), or RIP (4). In Cisco software, router processes are defined and redistributed on a per-VRF basis, not per-interface. Therefore, all interfaces assigned to the same VRF have the same value for this object.
mplsVpnInterfaceConfStorageType	Read-only implementation. This object always reads “volatile (2).”
mplsVpnInterfaceConfRowStatus	Read-only implementation. This object normally reads “active (1),” but may read “notInService (2),” if a VRF was recently deleted.

mplsVpnVrfRouteTargetTable

The route target table (mplsVpnVrfRouteTargetTable) describes the route target communities that are defined for a particular VRF. An LSR creates an entry in this table for each target configured for a VRF supporting an MPLS VPN instance.

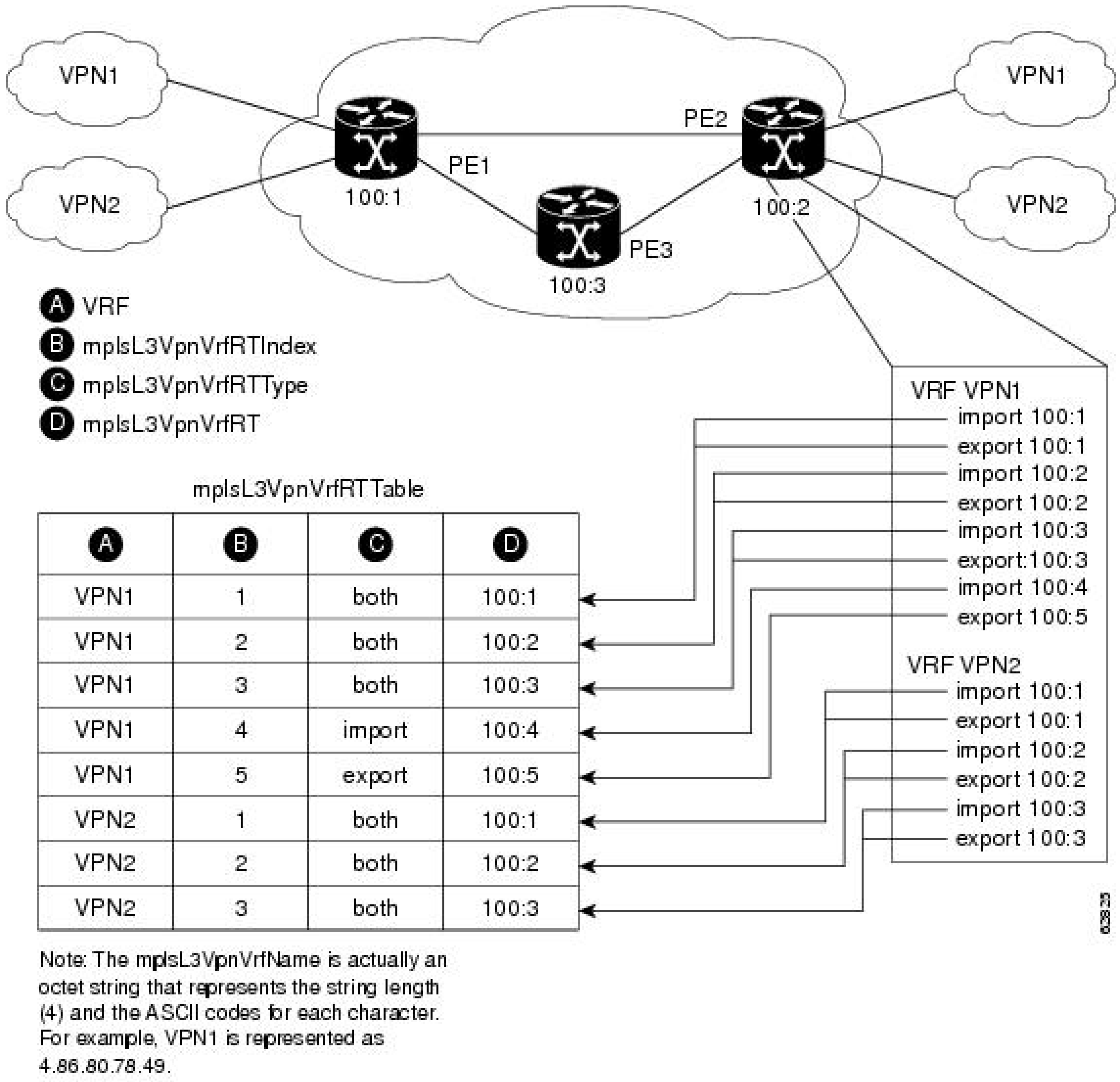
The distribution of VPN routing information is controlled through the use of VPN route target communities, implemented by BGP extended communities. Distribution of VPN routing information works as follows:

- When a VPN route learned from a customer edge (CE) router is injected into BGP, a list of VPN route target extended community attributes is associated with it. Typically the list of route target community values is set from an export list of route targets associated with the VRF from which the route was learned.

- An import list of route target extended communities is associated with each VRF. The import list defines route target extended community attributes a route must have for the route to be imported into the VRF. For example, if the import list for a particular VRF includes route target communities A, B, and C, then any VPN route that carries any of those route target extended communities--A, B, or C--is imported into the VRF.

The figure below shows a sample configuration and its relationship to an mplsVpnVrfRouteTargetTable. A route target table exists on each PE router. Routers with route distinguishers (RDs) 100:1, 100:2, and 100:3 are shown in the sample configuration. Routers with RDs 100:4 and 100:5 are not shown in the figure, but are included in the route targets for PE2 and in the mplsVpnVrfRouteTargetTable.

Figure 34: Sample Configuration and the mplsVpnVrfRouteTargetTable



A	B	C	D
VPN1	1	both	100:1
VPN1	2	both	100:2
VPN1	3	both	100:3
VPN1	4	import	100:4
VPN1	5	export	100:5
VPN2	1	both	100:1
VPN2	2	both	100:2
VPN2	3	both	100:3

Note: The mplsL3VpnVrfName is actually an octet string that represents the string length (4) and the ASCII codes for each character. For example, VPN1 is represented as 4.86.80.78.49.

The mplsVpnVrfRouteTargetTable shows the import and export route targets for each VRF. The table is indexed by the following:

- mplsVpnVrfName--The VRF name
- mplsVpnVrfRouteTargetIndex--The route target entry identifier
- mplsVpnVrfRouteTargetType--A value specifying whether the entry is an import route target, export route target, or is defined as both

The table below lists the MIB objects and their functions for this table.

Table 67: PPVPN-MPLS-VPN MIB Objects for the mplsVpnVrfRouteTargetTable

MIB Object	Function
mplsVpnVrfRouteTargetIndex	A value that defines each route target's position in the table.
mplsVpnVrfRouteTargetType	Determines which type of route target the entry represents: import (1), export (2), or both (3).
mplsVpnVrfRouteTarget	Determines the route distinguisher for this target.
mplsVpnVrfRouteTargetDescr	Description of the route target. This object is not supported. Therefore, the object is the same as mplsVpnVrfRouteTarget.
mplsVpnVrfRouteTargetRowStatus	Read-only implementation. This object normally reads "active (1)," but may read "notInService (2)," if a VRF was recently deleted.

mplsVpnVrfBgpNbrAddrTable

The BGP neighbor address table (mplsVpnVrfBgpNbrAddrTable) represents the MPLS external Border Gateway Protocol (eBGP) neighbors that are defined for a particular VRF. An LSR creates an entry for every BGP neighbor that is defined in the VRF's address-family.

The mplsVpnVrfBgpNbrAddrTable is indexed by the following:

- mplsVpnVrfName--The VRF name
- mplsVpnInterfaceConfIndex--An identifier that is the same as the ifIndex from the Interface MIB of the interface assigned to the VRF
- mplsVpnVrfBgpNbrIndex--The IP address of the neighbor

The table below lists the MIB objects and their functions for this table.

Table 68: PPVPN-MPLS-VPN MIB Objects for the mplsVpnVrfBgpNbrAddrTable

MIB Object	Function
mplsVpnVrfBgpNbrIndex	The IPv4 address of the eBGP neighbor.

MIB Object	Function
mplsVpnVrfBgpNbrRole	The role of this eBGP neighbor: customer edge (1) or provider edge (2). If the object mplsVpnInterfaceVpnClassification is CSC, then this value is provider edge (2); otherwise, this value is customer edge (1).
mplsVpnVrfBgpNbrType	Address type of this eBGP neighbor. The MIB supports only IPv4 (1). Therefore, this object returns “ipv4 (1).”
mplsVpnVrfBgpNbrAddr	IP address of the eBGP neighbor.
mplsVpnVrfBgpNbrRowStatus	Read-only implementation. This object normally reads “active (1),” but may read “notInService (2)” if a VRF was recently deleted.
mplsVpnVrfBgpNbrStorageType	Read-only implementation. This object always reads “volatile (2).”

mplsVpnVrfSecTable

The VRF security table (mplsVpnVrfSecTable) provides information about security for each VRF. An LSR creates an entry in this table for every VRF capable of supporting MPLS VPN.

The mplsVpnVrfSecTable *augments* the mplsVpnVrfTable and has the same indexing.

The table below lists the MIB objects and their functions for this table.

Table 69: PPVPN-MPLS-VPN MIB Objects for the mplsVpnVrfSecTable

MIB Object	Function
mplsVpnVrfSecIllegalLabelViolations	The number of illegally received labels on a VRF interface. Only illegal labels are counted by this object, therefore the object only applies to a VRF interface that is MPLS enabled (CSC situation). This counter is incremented whenever a label is received that is above or below the valid label range, not in the global label forwarding table, or is received on the wrong VRF (that is, table IDs for the receiving interface and appropriate VRF label forwarding table do not match).
mplsVpnVrfSecIllegalLabelRcvThresh	Notification threshold for illegal labels received on this VRF. When the number of illegal labels received on this interface crosses this threshold, an mplsNumVrfSecIllegalLabelThreshExceeded notification is sent (if the notification is enabled and configured). This object is one of the few in this MIB agent that supports the SNMP SET operation, which allows you to change this value.

mplsVpnVrfPerfTable

The VRF performance table (mplsVpnVrfPerfTable) provides statistical performance information for each VRF. An LSR creates an entry in this table for every VRF capable of supporting MPLS VPN.

The mplsVpnVrfPerfTable *augments* the mplsVpnVrfTable and has the same indexing.

The table below lists the MIB objects and their functions for this table.

Table 70: PPVPN-MPLS-VPN MIB Objects for the mplsVpnVrfPerfTable

MIB Objects	Functions
mplsVpnVrfPerfRoutesAdded	The number of routes added to this VRF over the course of its lifetime.
mplsVpnVrfPerfRoutesDeleted	The number of routes removed from this VRF.
mplsVpnVrfPerfCurrNumRoutes	The number of routes currently defined within this VRF.

mplsVpnVrfRouteTable

The VRF routing table (mplsVpnVrfRouteTable) provides the IP routing table information for each VRF. The information available in this table can also be accessed with the **show ip route vrf** *vrf-name* command. For example, for PE1 in the figure above:

- With the **show ip route vrf vpn1** command, you would see results like the following:

```
Router# show ip route vrf vpn1
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
!
Gateway of last resort is not set
!
     10.0.0.0/32 is subnetted, 3 subnets
B       10.3.0.0 [200/0] via 192.168.2.1, 04:36:33
C       10.1.0.0/16 is directly connected, FastEthernet1
C       10.2.0.0/16 [200/0] directly connected FastEthernet2, 04:36:33
```

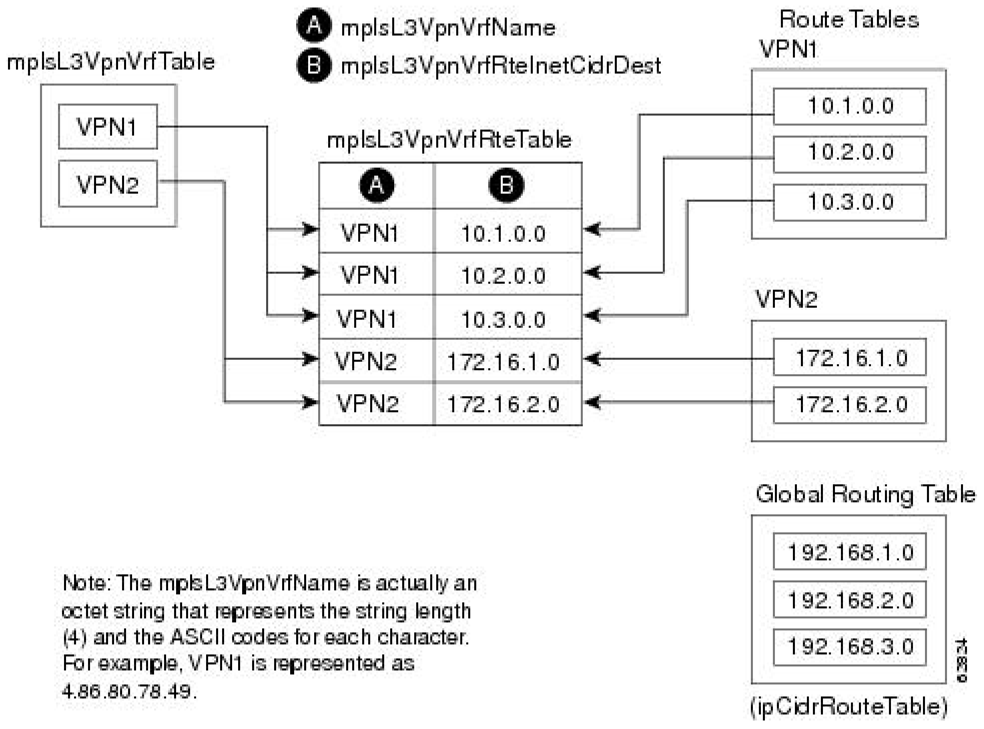
- With the **show ip route vrf vpn2** command, you would see results like the following:

```
Router# show ip route vrf vpn2
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
!
Gateway of last resort is not set
!
     172.16.0.0/32 is subnetted, 2 subnets
```

```
B       172.16.2.0 [200/0] via 192.168.2.1, 04:36:33
C       172.16.1.0 is directly connected, ATM 3/0
```

The figure below shows the relationship of the routing tables, the VRFs, and the mplsVpnVrfRouteTable. You can display information about the VPN1 and VPN2 route tables using the **show ip route vrf** *vrf-name* command. The global route table is the same as ipCidrRouteTable in the IP-FORWARD-MIB. You can display information about the global route table with the **show ip route** command.

Figure 35: Route Table, VRFs, and the mplsVpnVrfRouteTable



An LSR creates an entry in this table for every route that is configured, either dynamically or statically, within the context of a specific VRF capable of supporting MPLS VPN.

The mplsVpnVrfRouteTable is indexed by the following:

- mplsVpnVrfName--The VRF name, which provides the VRF routing context
- mplsVpnVrfRouteDest--The IP destination address
- mplsVpnVrfRouteMask--The IP destination mask
- mplsVpnVrfRouteTos--The IP header ToS bits
- mplsVpnVrfRouteNextHop--The IP address of the next hop for each route entry

Note The ToS bits are not supported and, therefore, are always 0.

The table below lists the MIB objects and their functions for the mplsVpnVrfRouteTable. This table represents VRF-specific routes. The global routing table is the ipCidrRouteTable in the IP-FORWARD-MIB.

Table 71: PPVPN-MPLS-VPN MIB Objects for the mplsVpnVrfRouteTable

MIB Object	Function
mplsVpnVrfRouteDest	The destination IP address defined for this route.
mplsVpnVrfRouteDestAddrType	The address type of the IP destination address (mplsVpnVrfRouteDest). This MIB implementation supports only IPv4 (1). Therefore, this object has a value of "ipv4 (1)."
mplsVpnVrfRouteMask	The destination IP address mask defined for this route.
mplsVpnVrfRouteMaskAddrType	The address type of the destination IP address mask. This MIB implementation supports only IPv4 (1). Therefore, this object has a value of "ipv4 (1)."
mplsVpnVrfRouteTos	The ToS bits from the IP header for this route. Cisco software supports only ToS bits of zero. Therefore, the object is always 0.
mplsVpnVrfRouteNextHop	The next hop IP address defined for this route.
mplsVpnVrfRouteNextHopAddrType	The address type of the next hop IP address. This MIB implementation only supports only IPv4 (1). Therefore, this object has a value of "ipv4 (1)."
mplsVpnVrfRouteIfIndex	The interface MIB ifIndex for the interface through which this route is forwarded. The object is 0 if no interface is defined for the route.
mplsVpnVrfRouteType	Defines if this route is a local or remotely defined route.
mplsVpnVrfRouteProto	The routing protocol that was responsible for adding this route to the VRF.
mplsVpnVrfRouteAge	The number of seconds since this route was last updated.
mplsVpnVrfRouteInfo	A pointer to more information from other MIBs. This object is not supported and always returns "nullOID (0.0)."
mplsVpnVrfRouteNextHopAS	The autonomous system number of the next hop for this route. This object is not supported and is always 0.
mplsVpnVrfRouteMetric1	The primary routing metric used for this route.
mplsVpnVrfRouteMetric2 mplsVpnVrfRouteMetric3 mplsVpnVrfRouteMetric4 mplsVpnVrfRouteMetric5	Alternate routing metrics used for this route. These objects are supported only for Cisco Interior Gateway Routing Protocol (IGRP) and Cisco Enhanced Interior Gateway Routing Protocol (EIGRP). These objects display the bandwidth metrics used for the route. Otherwise, these values are set to -1.

MIB Object	Function
mplsVpnVrfRouteRowStatus	Read-only implementation. This object normally reads “active (1),” but may read “notInService (2),” if a VRF was recently deleted.
mplsVpnVrfRouteStorageType	Read-only implementation. This object always reads “volatile (2).”

PPVPN-MPLS-VPN MIB Notifications

This section provides the following information about supported PPVPN-MPLS-VPN MIB notifications:

PPVPN-MPLS-VPN MIB Notification Events

The following notifications of the PPVPN-MPLS-VPN MIB are supported:

- mplsVrfIfUp--Sent to an NMS when an interface comes up and is assigned a VRF instance.
- mplsVrfIfDown--Generated and sent to the NMS when a VRF is removed from an interface or the interface transitions from an operationally “up” state to a “down” state.
- mplsNumVrfRouteMidThreshExceeded--Generated and sent when the middle (warning) threshold is crossed. You can configure this threshold in the CLI by using the following commands:

```
Router(config)# ip vrf vrf-name
Router(config-vrf)#  maximum routes limit warn-threshold (% of max)
```

The *warn-threshold* argument is a percentage of the maximum routes specified by the *limit* argument. You can also configure a middle threshold with the following command, in which the *limit* argument represents the warning threshold:

```
Router(config-vrf)# maximum routes limit warn-threshold (% of max)
```

This notification is sent to the NMS only at the time the threshold is exceeded. (See the figure below for a comparison of the warning and maximum thresholds.) Whenever the number of routes falls below this threshold and exceeds the threshold again, a notification is sent to the NMS.

- MplsNumVrfRouteMaxThreshExceeded--Generated and sent when you attempt to create a route on a VRF that already contains the maximum number of routes as defined by the *limit* argument of the **maximum routes**c ommands:

```
Router(config)# ip vrf vrf-name
Router(config-vrfmaximum routes limit warn-threshold (% of max)
```

A trap notification is sent to the NMS when you attempt to exceed the maximum threshold. Another MplsNumVrfRouteMaxThreshExceeded notification is not sent until the number of routes falls below the maximum threshold and reaches the maximum threshold again. (See the figure below for an example of how this notification works and for a comparison of the maximum and warning thresholds.)

Note The **maximum routes** command sets the number of routes for a VRF. You *cannot* exceed the number of routes in the VRF that you set with the **maximum routes** *limit warn-threshold* command. Prior to implementation of the PPVPN-MPLS-VPN MIB, you were not notified when this threshold (or the warning threshold) was reached.

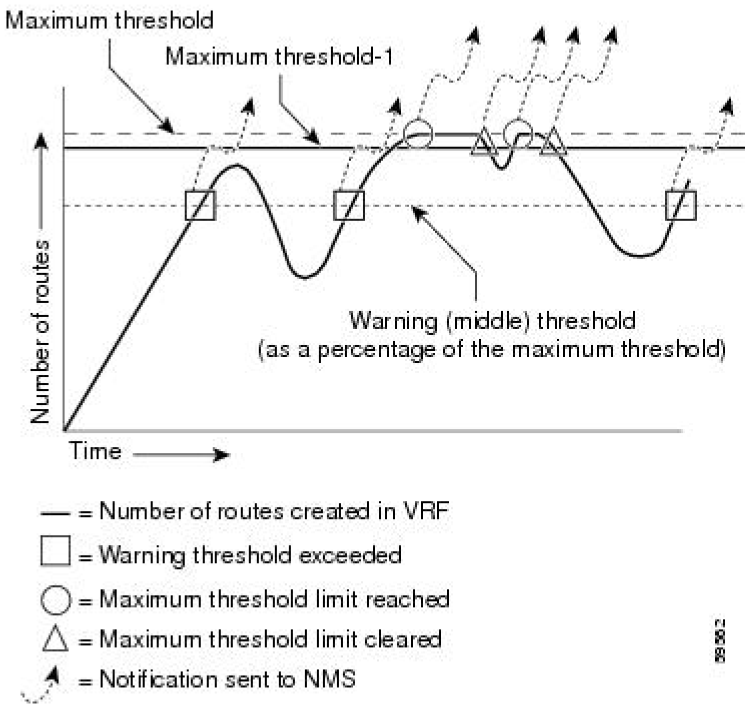
- mplsNumVrfSecIllegalLabelThreshExceeded--Generated and sent when the number of illegal labels received on a VRF interface exceeds the threshold *mplsVpnVrfSecIllegalLabelRcvThresh* . This threshold is defined with a value of 0. Therefore, a notification is sent when the first illegal label is received on a VRF. Labels are considered illegal if they are outside of the valid label range, do not have a Label Forwarding Information Base (LFIB) entry, or the table ID of the message does not match the table ID for the label in the LFIB.

CISCO-IETF-PPVPN-MPLS-VPN MIB Notification Events

The following notification of the CISCO-IETF-PPVPN-MPLS-VPN MIB is supported in Cisco software:

- cMplsNumVrfRouteMaxThreshCleared--Generated and sent when the number of routes on a VRF attempts to exceed the maximum number of routes and then drops below the maximum number of routes. If you attempt to create a route on a VRF that already contains the maximum number of routes, the mplsNumVrfRouteMaxThreshExceeded notification is sent (if enabled). When you remove routes from the VRF so that the number of routes falls below the set limit, the cMplsNumVrfRouteMaxThreshCleared notification is sent. You can clear all routes from the VRF by using the **clear ip route vrf** command. (See the figure below to see when the cMplsNumVrfRouteMaxThreshCleared notification is sent.)

Figure 36: Comparison of Warning and Maximum Thresholds


Maximum threshold
Maximum threshold-1
Number of routes
Warning (middle) threshold
(as a percentage of the maximum threshold)
Time
= Number of routes created in VRF
= Warning threshold exceeded
= Maximum threshold limit reached
= Maximum threshold limit cleared
= Notification sent to NMS
59562

Notification Specification

In an SNMPv1 notification, each VPN notification has a generic type identifier and an enterprise-specific type identifier for identifying the notification type.

- The generic type for all VPN notifications is “enterpriseSpecific” because this is not one of the generic notification types defined for SNMP.
- The enterprise-specific type is identified as follows:
  - 1 for *mplsVrfIfUp*
  - 2 for *mplsVrfIfDown*
  - 3 for *mplsNumVrfRouteMidThreshExceeded*
  - 4 for *mplsNumVrfRouteMaxThreshExceeded*
  - 5 for *mplsNumVrfSecIllegalLabelThreshExceeded*
  - 6 for cMplsNumVrfRouteMaxThreshCleared

In SNMPv2, the notification type is identified by an SnmpTrapOID varbind (variable binding consisting of an object identifier [OID] type and value) included within the notification message.

Each notification also contains two additional objects from the PPVPN-MPLS-VPN MIB. These objects provide additional information about the event, as follows:

- The VRF interface up/down notifications provide additional variables--*mplsVpnInterfaceConfIndex* and *mplsVpnVrfName*-- in the notification. These variables describe the SNMP interface index and the VRF name, respectively.
- The mid and max threshold notifications include the *mplsVpnVrfName* variable (VRF name) and the *mplsVpnVrfPerfCurrNumRoutes* variable that indicates the current number of routes within the VRF.
- The illegal label notification includes the *mplsVpnVrfName* variable (VRF name) and the *mplsVpnVrfSecIllegalLabelViolations* variable that maintains the current count of illegal labels on a VPN.

Monitoring the PPVPN-MPLS-VPN MIB Notifications

When PPVPN-MPLS-VPN MIB notifications are enabled (see the **snmp-server enable traps mpls vpn** command in the Cisco IOS Multiprotocol Label Switching Command Reference), notification messages relating to specific MPLS VPN events within Cisco software are generated and sent to a specified NMS in the network. Any utility that supports SNMPv1 or SNMPv2 notifications can receive notification messages.

To monitor PPVPN-MPLS-VPN MIB notification messages, log in to an NMS that supports a utility that displays SNMP notifications, and start the display utility.

Unsupported Objects in PPVPN-MPLS-VPN MIB

The following objects from the mplsVpnVrfBgpPathAttrTable are not supported with SNMP management for MPLS VPN features in Cisco software:

- mplsVpnVrfBgpPathAttrPeer

- mplsVpnVrfBgpPathAttrIpAddrPrefixLen
- mplsVpnVrfBgpPathAttrIpAddrPrefix
- mplsVpnVrfBgpPathAttrOrigin
- mplsVpnVrfBgpPathAttrASPathSegment
- mplsVpnVrfBgpPathAttrNextHop
- mplsVpnVrfBgpPathAttrMultiExitDisc
- mplsVpnVrfBgpPathAttrLocalPref
- mplsVpnVrfBgpPathAttrAtomicAggregate
- mplsVpnVrfBgpPathAttrAggregatorAS
- mplsVpnVrfBgpPathAttrAggregatorAddr
- mplsVpnVrfBgpPathAttrCalcLocalPref
- mplsVpnVrfBgpPathAttrBest
- mplsVpnVrfBgpPathAttrUnknown

How to Configure MPLS VPN--MIB Support

Configuring the SNMP Community

An SNMP community string defines the relationship between the SNMP manager and the agent. The community string acts like a password to regulate access to the agent on the router.

Perform this task to configure an SNMP community.

SUMMARY STEPS

1. **enable**
2. **show running-config** [*options*]
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** [*options*]

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** [*options*] **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`	Sets up 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-nameview-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 only able to 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`
		Saves the modified configuration to NVRAM as the startup configuration file.
		• The **do** command allows you to perform EXEC level commands in configuration mode.

Step 6	**exit** **Example:** `Router(config)# exit`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 7	**show running-config** [*options*] **Example:** `Router# show-running config | include smnp-server`	(Optional) Displays the configuration information currently on the router, the configuration for a specific interface, or map-class information. • Use the **show running-config** command to check that the **snmp-server** statements appear in the output.

Configuring the Router to Send SNMP Traps

Perform this task to configure the router to sendm SNMP traps to a host.

The **snmp-server host** command specifies which hosts receive 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.

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-addr* [**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 vpn** [**illegal-label**] [**max-thresh-cleared**] [**max-threshold**] [**mid-threshold**] [**vrf-down**] [**vrf-up**]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server host** *host-addr* [**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-vpn`	Specifies the recipient of an SNMP notification operation. • The*host-addr* 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 and argument pair names the User Datagram Protocol (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. • The **vrf** *vrf-name* keyword and argument pair specifies the VRF table that should be used to send SNMP notifications.
Step 4	**snmp-server enable traps mpls vpn** [**illegal-label**] [**max-thresh-cleared**] [**max-threshold**] [**mid-threshold**] [**vrf-down**] [**vrf-up**] **Example:** `Router(config)# snmp-server enable traps mpls vpn vrf-down vrf-up`	Enables the router to send MPLS VPN-specific SNMP notifications (traps and informs). • The **illegal-label** keyword enables a notification for any illegal labels received on a VRF interface. Labels are illegal if they are outside the legal range, do not have an LFIB entry, or do not match table IDs for the label. • The **max-thresh-cleared** keyword enables a notification when the number of routes falls below the limit after the maximum route limit was attempted. • The **max-threshold** keyword enables a notification that a route creation attempt was unsuccessful because the maximum route limit was reached. Another MplsNumVrfRouteMaxThreshExceeded notification is not sent until the number of routes falls below the maximum threshold and reaches the maximum threshold

	Command or Action	Purpose
		again. The max-threshold value is determined by the **maximum routes** command in VRF configuration mode. • The **mid-threshold** keyword enables a notification of a warning that the number of routes created has crossed the warning threshold. This warning is sent only at the time the warning threshold is exceeded. • The **vrf-down** keyword enables a notification for the removal of a VRF from an interface or the transition of an interface to the down state. • The **vrf-up** keyword enables a notification for the assignment VRF to an interface that is operational or for the transition of a VRF interface to the operationally up state.

Step 5	**end**
	
	Example:
	
	`Router(config)# end`
		(Optional) Exits to privileged EXEC mode.

Configuring Threshold Values for MPLS VPN--SNMP Notifications

Perform this task to configure the following threshold values for MPLS VPN--SNMP notifications:

- The mplsNumVrfRouteMidThreshExceeded notification event is generated and sent when the middle (warning) threshold is crossed. You can configure this threshold in the CLI by using the **maximum routes** command in VRF configuration mode. This notification is sent to the NMS only at the time the threshold is exceeded. Whenever the number of routes falls below this threshold and exceeds the threshold again, a notification is sent to the NMS.
- The mplsNumVrfRouteMaxThreshExceeded notification event is generated and sent when you attempt to create a route on a VRF that already contains the maximum number of routes as defined by the **maximum routes** command in VRF configuration mode. A trap notification is sent to the NMS when you attempt to exceed the maximum threshold. Another MplsNumVrfRouteMaxThreshExceeded notification is not sent until the number of routes falls below the maximum threshold and reaches the maximum threshold again.

See the figure above for an example of how this notification works and for a comparison of the maximum and warning thresholds.

Note The **maximum routes** command sets the number of routes for a VRF. You *cannot* exceed the number of routes in the VRF that you set with the **maximum routes** *limit warn-threshold* command. Prior to the implementation of the PPVPN-MPLS-VPN MIB, you were not notified when this threshold (or the warning threshold) was reached.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **maximum routes** *limit* {*warn-threshold* | **warn-only**}
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	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf vpn1`	Configures a VRF routing table and enters VRF configuration mode. • The *vrf-name* argument specifies the name assigned to a VRF.

Step 4	**maximum routes** *limit* {*warn-threshold* | **warn-only**}
	
	Example:
	`Router(config-vrf)# maximum routes 10000 80`
		Limits the maximum number of routes in a VRF to prevent a PE router from importing too many routes.
		• The *limit* argument specifies the maximum number of routes allowed in a VRF. The range is from 1 to 4,294,967,295.
		• The *warn-threshold* argument generates a warning when the number of routes set by the *warn-threshold* argument is reached and rejects routes that exceed the maximum number set in the *limit* argument. The warning threshold is a percentage from 1 to 100 of the maximum number of routes specified in the *limit* argument.
		• The **warn-only** keyword specifies that a system logging error message is issued when the maximum number of routes allowed for a VRF exceeds the limit threshold. However, additional routes are still allowed.

Step 5	**end**
	
	Example:
	`Router(config-vrf)# end`
		(Optional) Exits to privileged EXEC mode.

Configuration Examples for MPLS VPN--MIB Support

Example Configuring the SNMP Community

The following example shows enabling a simple SNMP community group. This configuration permits any SNMP client to access all PPVPN-MPLS-VPN MIB objects with read-only access using the community string comaccess.

```
Router# configure terminal
Router(config)# snmp-server community comaccess ro
```

Verify that the SNMP master agent is enabled for the MPLS VPN--MIB Support feature:

```
Router# show running-config | include snmp-server
Building configuration...
.
snmp-server community comaccess RO
```

Note If you do not see any “snmp-server” statements, SNMP is not enabled on the router.

Example Configuring the Router to Send SNMP Traps

The following example shows you how to enable the router to send MPLS VPN notifications to host 172.20.2.160 using the comaccess community string if a VRF transitions from an up or down state:

```
Router# configure terminal
Router(config)# snmp-server host 172.20.2.160 traps comaccess mpls-vpn
Router(config)# snmp-server enable traps mpls vpn vrf-down vrf-up
```

Example Configuring Threshold Values for MPLS VPN--SNMP Notifications

The following example shows how to set a maximum threshold of 10,000 routes and a warning threshold that is 80 percent of the maximum threshold for a VRF named vpn1 on a router:

```
Router(config)# ip vrf vpn1
Router(config-vrf)# maximum routes 10000 80
```

The following example shows how to set a warning threshold of 10,000 routes for a VRF named vpn2 on a router. An error message is generated; however, additional routes are still allowed because a maximum route threshold is not set with this command.

```
Router(config)# ip vrf vpn2
Router(config-vrf)# maximum routes 10000 warn-only
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Description of commands associated with MPLS and MPLS applications	*Cisco IOS Multiprotocol Label Switching Command Reference*
MPLS VPN configuration tasks	Configuring MPLS Layer 3 VPNs
A description of SNMP agent support in Cisco software for the MPLS Traffic Engineering MIB (MPLS TE MIB)	MPLS Traffic Engineering (TE) MIB
Overview and configuration tasks for the MPLS distribution protocol	MPLS Label Distribution Protocol

Standards

Standard	Title
draft-ietf-ppvpn-mpls-vpn-mib-05	*MPLS/BGP Virtual Private Network Management Information Base Using SMIv2*

MIBs

MIB	MIBs Link
• MPLS-VPN-MIB • CISCO-IETF-PPVPN-MPLS-VPN-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 2233	*The Interfaces Group MIB using SMIv2*
RFC 2547	*BGP/MPLS VPNs*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for MPLS VPN--MIB Support

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 72: Feature Information for MPLS VPN--MIB Support

Feature Name	Releases	Feature Information
MPLS VPN--MIB Support	12.0(21)ST 12.0(22)S 12.2(13)S 12.2(15)T 12.0(24)S1 12.0(25)S 12.0(30)S 12.2(28)SB 12.2(33)SRA 12.2(31)SB2 12.2(33)SXH 12.4(20)T	This feature was introduced into Cisco IOS Release 12.0(21)ST. This feature was integrated into Cisco IOS Release 12.0(22)S. This feature was integrated into Cisco IOS Release 12.2(13)S. The PPVPN-MPLS-VPN MIB notifications were supported in Cisco IOS Release 12.2(13)T. The PPVPN-MPLS-VPN MIB tables were integrated into Cisco IOS Release 12.2(15)T. The feature was implemented for ATM and Frame Relay subinterfaces and integrated into Cisco IOS Release 12.0(24)S1. This feature was integrated into Cisco IOS Release 12.0(25)S. This feature was updated with the MPLS VPN Trap Enhancement feature, which introduced the cMplsNumVrfRouteMaxThreshCleared notification. The **max-thresh-cleared** keyword was added to the **snmp-server enable traps mpls vpn** command. This feature was integrated into Cisco IOS Release 12.2(28)SB. This feature was integrated into Cisco IOS Release 12.2(33)SRA. This feature was implemented into Cisco IOS Release 12.2(31)SB2. This feature was implemented into Cisco IOS Release 12.2(33)SXH. This feature was integrated into Cisco IOS Release 12.4(20)T.

Glossary

autonomous system --A collection of networks that share the same routing protocol and that are under the same system administration.

ASN.1 --Abstract Syntax Notation One. The data types independent of particular computer structures and representation techniques. Described by ISO International Standard 8824.

BGP --Border Gateway Protocol. The exterior Border Gateway Protocol used to exchange routing information between routers in separate autonomous systems. BGP uses TCP. Because TCP is a reliable protocol, BGP does not experience problems with dropped or fragmented data packets.

BGP prefixes--A route announcement using the BGP. A prefix is composed of a path of autonomous system numbers, indicating which networks the packet must pass through, and the IP block that is being routed. A BGP prefix would look something like: 701 1239 42 206.24.14.0/24. (The /24 part is referred to as a CIDR mask.) The /24 indicates that there are 24 ones in the netmask for this block starting from the left side. A /24 corresponds to the natural mask 255.255.255.0.

Cisco Express Forwarding --An advanced Layer 3 IP switching technology. Cisco Express Forwarding optimizes network performance and scalability for networks with large and dynamic traffic patterns.

CE router--customer edge router. A router on the border between a VPN provider and a VPN customer that belongs to the customer.

CIDR --classless interdomain routing. A technique supported by BGP4 and based on route aggregation. CIDR allows routers to group routes to reduce the quantity of routing information carried by the core routers. With CIDR, several IP networks appear to networks outside the group as a single, larger entity. With CIDR, IP addresses and their subnet masks are written as four octets, separated by periods, followed by a forward slash and a two-digit number that represents the subnet mask.

community --In SNMP, a logical group of managed devices and NMSs in the same administrative domain.

community name --See community string.

community string --A text string that acts as a password and is used to authenticate messages sent between a managed station and a router containing an SNMP agent. The community string is sent in every packet between the manager and the client. Also called a community name.

IETF --Internet Engineering Task Force. A task force consisting of over 80 working groups responsible for developing Internet standards. The IETF operates under the auspices of ISOC. *See also* ISOC.

informs --A type of notification message that is more reliable than a conventional trap notification message, because the informs message notification requires acknowledgment, and a trap notification does not.

ISOC --Internet Society. An international nonprofit organization, founded in 1992, that coordinates the evolution and use of the Internet. In addition, ISOC delegates authority to other groups related to the Internet, such as the IAB. ISOC is headquartered in Reston, Virginia (United States).

label --A short, fixed-length data construct that tells switching nodes how to forward data (packets or cells).

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. In the Cisco Label Switching system, the data structure for storing information about incoming and outgoing tags (labels) and associated equivalent packets suitable for labeling.

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

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

MPLS --Multiprotocol Label Switching. 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.

MPLS interface --An interface on which MPLS traffic is enabled.

MPLS VPN --Multiprotocol Label Switching Virtual Private Network. An IP network infrastructure delivering private network services over a public infrastructure using a Layer 3 backbone. Using MPLS VPNs in a Cisco network provides the capability to deploy and administer scalable Layer 3 VPN backbone services including applications, data hosting network commerce, and telephony services to business customers.

For an MPLS VPN solution, an MPLS VPN is a set of provider edge routers that are connected by means of a common "backbone" network to supply private IP interconnectivity between two or more customer sites for a given customer. Each VPN has a set of provisioning templates and policies and can span multiple provider administrative domains (PADs).

NMS --network management system. A powerful, well-equipped computer (typically an engineering workstation) that is used by a network administrator to communicate with other devices in the network. An NMS is typically used to manage network resources, gather statistics, and perform a variety of network administration and configuration tasks.

notification --A message sent by an SNMP agent to a network management station, console, or terminal to indicate that a significant event within Cisco software has occurred. *See also* trap.

PE router --provider edge router. A router on the border between a VPN provider and a VPN customer that belongs to the provider.

PPVPN --Provider-Provisioned VPN. The name of the IETF working group that is developing the PPVPN-MPLS-VPN MIB.

QoS --quality of service. A measure of performance for a transmission system that reflects its transmission quality and service availability.

RSVP --Resource Reservation Protocol. A protocol for reserving network resources to provide quality of service guarantees to application flows.

RT --route target. An extended community attribute that identifies a group of routers and, in each router of that group, a subset of forwarding tables maintained by the router that can be populated with a BGP route carrying that extended community attribute. The RT is a 64-bit value by which Cisco discriminates routes for route updates in VRFs.

SNMP --Simple Network Management Protocol. The network management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, and to manage configurations, statistics collection, performance, and security. *Seealso* SNMP2.

SNMP2 --SNMP Version 2. Version 2 of the popular network management protocol. SNMP2 supports centralized and distributed network management strategies, and includes improvements in the Structure of Management Information (SMI), protocol operations, management architecture, and security. *See also* SNMP.

traffic engineering --The techniques and processes used to cause routed traffic to travel through the network on a path other than the one that would have been chosen if standard routing methods had been used.

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

VPN --Virtual Private Network. A group of sites that, as the result of a set of administrative policies, are able to communicate with each other over a shared backbone network. A VPN is a secure IP-based network that shares resources on one or more physical networks. A VPN contains geographically dispersed sites that can communicate securely over a shared backbone. See alsoMPLS VPN.

VPN ID --A mechanism that identifies a VPN based on RFC 2685. A VPN ID consists of an Organizational Unique Identifier (OUI), a three-octet hex number assigned by the IEEE Registration Authority, and a VPN index, a four-octet hex number, which identifies the VPN within the company.

VRF --VPN routing and forwarding instance. A VRF consists of an IP routing table, a derived forwarding table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols that determine what goes into the forwarding table. In general, a VRF includes the routing information that defines a customer VPN site that is attached to a PE router.

CHAPTER 11

MPLS VPN SNMP Notifications

This document describes the Simple Network Management Protocol (SNMP) agent support in Cisco IOS for Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) event notifications as implemented in the notifications section of the draft *MPLS/BGP Virtual Private Network Management Information Base Using SMIv2* (*draft-ietf-ppvpn-mpls-vpn-mib-03.txt*).

The MPLS VPN technology allows service providers to offer intranet and extranet VPN services that directly connect their customers' remote offices to a public network with the same security and service levels that a private network offers. The Provider-Provisioned VPN (PPVPN)-MPLS-VPN MIB notifications provide SNMP notification for critical MPLS VPN events.

The MPLS VPN SNMP Notifications feature provides the following benefits:

- A standards-based SNMP interface for retrieving information about critical MPLS VPN events.
- The generation and queuing of notifications that call attention to major changes in the operational status of MPLS VPN enabled interfaces; the forwarding of notification messages to a designated NMS for evaluation and action by network administrators.
- Advanced warning when VPN routing tables are approaching or exceed their capacity.
- Warnings about the reception of illegal labels on a VRF enabled interface. Such receptions may indicate misconfiguration or an attempt to violate security.

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 VPN SNMP Notifications

The MPLS VPN SNMP Notifications feature requires the following:

- SNMP is installed and enabled on the label switching routers.
- Multiprotocol Label Switching (MPLS) is enabled on the label switching routers.
- Multiprotocol Border Gateway Protocol (BGP) is enabled on the label switching routers.
- Cisco Express Forwarding is enabled on the label switching routers.

Restrictions for MPLS VPN SNMP Notifications

- The MPLS-VPN-MIB agent is not implemented in this release.
- Configuration of the MIB using the SNMP SET command is not supported in this release.
- The retrieval of MPLS-VPN-MIB objects using SNMP GET is not supported in this release.

Information About MPLS VPN SNMP Notifications

Cisco Implementation of MPLS VPN MIB

SNMP agent code operating with the notifications of the MPLS VPN SNMP Notifications feature enables a standardized, SNMP-based approach to monitoring the MPLS VPN MIB notifications that aid in the management of Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs) in Cisco software.

The MPLS VPN SNMP Notifications feature is based on the IETF draft specification *draft-ietf-ppvpn-mpls-vpn-mib-02.txt*, which includes notification objects that support MPLS VPN notification events. This IETF draft MIB, which undergoes revisions from time to time, is being evolved toward becoming a standard. Accordingly, the Cisco implementation of features of the MPLS VPN MIB is expected to track the evolution of the IETF draft MIB, and may change accordingly.

Some slight differences between the IETF draft MIB and the actual implementation of MPLS VPNs within Cisco software require some minor translations between the MPLS VPN MIB and the internal data structures of Cisco software. These translations are accomplished by means of the SNMP agent code. Also, while running as a low priority process, the SNMP agent provides a management interface to Cisco software. SNMP adds little overhead on the normal functions of the device.

The SNMP objects defined in the MPLS VPN MIB notifications can be viewed by any standard SNMP utility. The network administrator can retrieve information in the MPLS VPN MIB using standard SNMP **get** and **getnext** operations for SNMP v1, v2, and v3.

All MPLS VPN MIB objects are based on the IETF draft MIB; thus, no specific Cisco SNMP application is required to support the functions and operations pertaining to the MPLS VPN SNMP Notifications feature.

This section contains the following information about the Cisco implementation of the MPLS VPN MIB:

Capabilities Supported by MPLS VPN SNMP Notifications

The following functionality is supported in this release for the MPLS VPN SNMP Notifications feature. This feature provides you with the ability to do the following:

- Create and send notification messages that signal changes when critical Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) events occur.
- Enable, disable, and configure notification messages for MPLS VPN events by using extensions to existing SNMP CLI commands.
- Specify the IP address of a network management system (NMS) in the operating environment to which notification messages are sent.
- Write notification configurations into nonvolatile memory.

Notification Generation Events for the MPLS VPN MIB

The following notifications of the MPLS VPN MIB are implemented for this release:

- **mplsVrfIfUp—**Sent to an NMS when an interface comes up and is assigned a VPN routing/forwarding table instance (VRF).
- **mplsVrfIfDown—**Generated and sent to the NMS when a VRF is removed from an interface or the interface transitions from an operationally "up" state to a "down" state.

Note For the mplsVrfIfUp or mplsVrfIfDown notifications to be issued on ATM or Frame Relay subinterfaces, you must configure the **snmp-server traps atm subif** command or the **snmp-server traps frame-relay subif** command on the subinterfaces, respectively.

- **mplsNumVrfRouteMidThreshExceeded—**Generated and sent when the middle (warning) threshold is crossed. You can configure this threshold in the CLI by using the following commands:

```
Router(config)# ip vrf vrf-name
Router(config-vrf)# maximum routes max-thresh
mid-thresh (% of max)
```

This notification is sent to the NMS only at the time the threshold is exceeded. Whenever the number of routes falls below this threshold and exceeds the threshold again, a notification is sent to the NMS. (See the figure below for a comparison of the warning and maximum thresholds.)

- **mplsNumVrfRouteMaxThreshExceeded—**Generated and sent when you attempt to create a route on a VRF that already contains the maximum number of routes as defined by the following CLI commands:

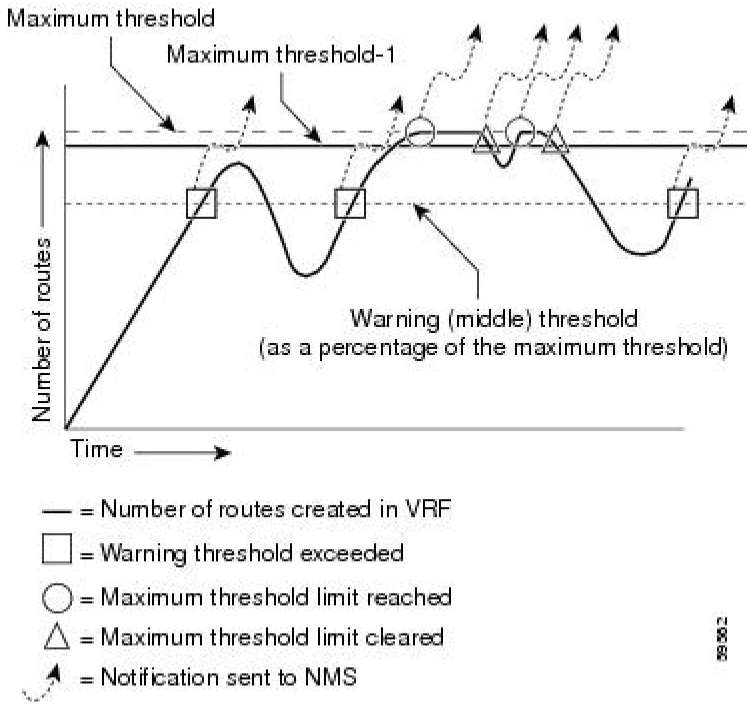
```
Router(config)# ip vrf vrf-name
Router(config-vrf)# maximum routes max-thresh
mid-thresh (% of max)
```

A trap notification is sent to the NMS when you attempt to exceed the maximum threshold. Another notification is not sent until the number of routes falls below the maximum threshold and reaches the maximum threshold again. (See the figure below for an example of how this notification works and for a comparison of the maximum and warning thresholds.)

Note The **maximum routes** command sets the number of routes for a VRF. You *cannot* exceed the number of routes in the VRF that you set with the **maximum routes** *max-thresh* CLI command. Prior to this implementation of the MPLS-VPN-MIB, you were not notified when this threshold (or the warning threshold) was reached.

- **mplsNumVrfSecIllegalLabelThreshExceeded—**Generated and sent when the amount of illegal labels received on a VRF interface exceeds the threshold *mplsVpnVrfSecIllegalLabelRcvThresh*. This threshold is defined with a value of 0. Therefore, a notification is sent when the first illegal label is received on a VRF. Labels are considered illegal if they are outside of the valid label range, do not have a Label Forwarding Information Base (LFIB) entry, or the table ID of the message does not match the table ID for the label in the LFIB.

Figure 37: Comparison of Warning and Maximum Thresholds



Notification Specification for MPLS-VPN-MIB

In an SNMPv1 notification, each VPN notification has a generic type identifier and an enterprise-specific type identifier for identifying the notification type.

- The generic type for all VPN notifications is "enterpriseSpecific" as this is not one of the generic notification types defined for SNMP.
- The enterprise-specific type is identified as follows:
    - 1 for *mplsVrfIfUp*
    - 2 for *mplsVrfIfDown*
    - 3 for *mplsNumVrfRouteMidThreshExceeded*
    - 4 for *mplsNumVrfRouteMaxThreshExceeded*
    - 5 for *mplsNumVrfSecIllegalLabelThreshExceeded*

In SNMPv2, the notification type is identified by an **SnmpTrapOID** varbind (variable binding consisting of an object identifier (OID) type and value) included within the notification message.

Each notification also contains two additional objects from the MPLS-VPN-MIB. These objects provide additional information about the event, as follows:

- The VRF interface up/down notifications provide additional variables--*mplsVpnInterfaceConfIndex* and *mplsVpnVrfName*-- in the notification. These variables describe the SNMP interface index and the VRF name, respectively.
- The mid and max threshold notifications include the *mplsVpnVrfName* variable (VRF name) as well as the *mplsVpnVrfPerfCurrNumRoutes* variable that indicates the current number of routes within the VRF.
- The illegal label notification includes the *mplsVpnVrfName* variable (VRF name) and the *mplsVpnVrfSecIllegalLabelViolations* variable that maintains the current count of illegal labels on a VPN.

Monitoring the MPLS VPN SNMP Notifications

When MPLS-VPN-MIB notifications are enabled, notification messages relating to specific Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) events within Cisco software are generated and sent to a specified network management system (NMS) in the network. Any utility that supports SNMPv1 or SNMPv2 notifications can receive notification messages.

To monitor MPLS-VPN-MIB notification messages, log in to an NMS that supports a utility that displays SNMP notifications, and start the display utility.

How to Configure the MPLS VPN SNMP Notifications

Configuring an SNMP Community

An SNMP community string defines the relationship between the SNMP manager and the agent. The community string acts like a password to regulate access to the agent on the device.

Perform this task to configure an SNMP community.

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**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show running-config** **Example:** `Device# 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:** `Device# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [*acl-number*] **Example:** `Device(config)# snmp-server community comaccess ro`	Sets up the community access string to permit access to the Simple Network Management Protocol (SNMP). • The *string* argument acts like a password and permits access to the SNMP protocol. • The **view***view-name* keyword and argument specifies the name of a previously defined view. The view defines the objects available to the community.

	Command or Action	Purpose
		• The **ro** keyword specifies read-only access. Authorized management stations are only able to 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:
	
	`Device(config)# do copy running-config startup-config`
		Saves the modified configuration to nonvolatile memory (NVRAM) as the startup configuration file. (The **do** command allows you to perform Exec level commands in configuration mode.)

Configuring the Device to Send SNMP Traps

Perform this task to configure the device to send traps to a host.

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

Note Although you can set the *community-string* argument using the **snmp-server host** command by itself, we recommend 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-addr* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]
4. Do one of the following:
   - **snmp-server enable traps atm** [**pvc** | **subif**]
   - **snmp-server enable traps frame-relay** [**subif**]
5. **snmp-server enable traps mpls vpn**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server host** *host-addr* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] **Example:** `Device(config)# snmp-server host 172.20.2.160 traps comaccess mpls-vpn`	Specifies the recipient of an SNMP notification operation. • The *host-addr* 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, as 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 and argument 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. MPLS VPN notifications are specified with the **mpls-vpn** keyword.
Step 4	Do one of the following: • **snmp-server enable traps atm** [**pvc** | **subif**] • **snmp-server enable traps frame-relay** [**subif**]	(For ATM subinterfaces only) Enables the sending of ATM SNMP notifications. • The **pvc** keyword enables SNMP ATM permanent virtual circuit (PVC) traps. • The **subif** keyword enables SNMP ATM subinterface traps. or

	Command or Action	Purpose

	Example:
	`Device(config)# snmp-server enable traps atm subif`
	Example:
	`Device(config)# snmp-server enable traps frame-relay subif`
		(For Frame Relay subinterfaces only) Enables Frame Relay DLCI link status SNMP notifications.
		• The **subif** keyword enables SNMP Frame Relay subinterface traps.
		Note For mplsVrfIfUp or mplsVrfIfDown notifications to be issued on ATM or Frame Relay subinterfaces, you must configure the appropriate **snmp-server enable traps** command with the **subif** keyword.

Step 5	**snmp-server enable traps mpls vpn** **Example:** `Device(config)# snmp-server enable traps mpls vpn vrf-up vrf-down`	Enables the device to send MPLS VPN SNMP notifications.

Step 6	**end**
	Example:
	`Device(config)# end`
		(Optional) Returns to user EXEC mode.

Configuring Threshold Values for MPLS VPN SNMP Notifications

Perform this task to configure threshold values for MPLS VPN SNMP notifications.

The **mplsNumVrfRouteMidThreshExceeded** notification event is generated and sent when the middle (warning) threshold is crossed. You can configure this threshold in the CLI by using the **maximum routes** command in VRF configuration mode. This notification is sent to the NMS only at the time the threshold is exceeded. Whenever the number of routes falls below this threshold and exceeds the threshold again, a notification is sent to the NMS.

The **mplsNumVrfRouteMaxThreshExceeded** notification event is generated and sent when you attempt to create a route on a VRF that already contains the maximum number of routes as defined by the **maximum routes** command in VRF configuration mode. A trap notification is sent to the NMS when you attempt to exceed the maximum threshold. Another notification is not sent until the number of routes falls below the maximum threshold and reaches the maximum threshold again.

(See the figure above for an example of how this notification works and for a comparison of the maximum and warning thresholds.)

Note The **maximum routes** command sets the number of routes for a VRF. You *cannot* exceed the number of routes in the VRF that you set with the **maximum routes** *max-thresh* CLI command. Prior to this implementation of the MPLS-VPN-MIB, you were not notified when this threshold (or the warning threshold) was reached.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **maximum routes** *limit* {*warn-threshold* | **warning-only**}
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Device(config)# ip vrf vpn1`	Configures a VRF routing table. • The *vrf-name* argument specifies the name assigned to a VRF.

Step 4	**maximum routes** *limit* {*warn-threshold* | **warning-only**}
	
	Example:
	`Device(config-vrf)# maximum routes 10000 80`
		Limits the maximum number of routes in a VRF to prevent a PE device from importing too many routes.
		• The *limit* argument specifies the maximum number of routes allowed in a VRF. You may select from 1 to 4,294,967,295 routes to be allowed in a VRF.
		• The *warn-threshold* argument specifies when the threshold limit is reached and routes are rejected. The threshold limit is a percentage of the *limit* specified, from 1 to 100 percent.
		• The **warning-only** keyword specifies that a SYSLOG error message is issued when the maximum number of routes allowed for a VRF exceeds the threshold. However, additional routes are still allowed.

	Command or Action	Purpose

Step 5	**end**
	
	Example:
	
	`Device(config-vrf)# end`
		(Optional) Returns to privileged EXEC mode.

Configuration Examples for MPLS VPN SNMP Notifications

Example: Configuring the Community

The following example shows enabling a simple SNMP community group. This configuration permits any SNMP client to access all MPLS-VPN-MIB objects with read-only access using the community string comaccess.

```
Device# configure terminal
Device(config)# snmp-server community comaccess ro
```

Verify that the SNMP master agent is enabled for the MPLS VPN SNMP Notifications feature:

```
Device# show running-config | include snmp-server
Building configuration...
....
snmp-server community comaccess RO
....
```

Note If you do not see any “snmp-server” statements, SNMP has not been enabled on the device.

Example: Configuring the Device to Send SNMP Traps

The following example shows you how to enable the device to send MPLS VPN notifications to host 172.20.2.160 using the comaccess community string if a VRF transitions from a down state to an up state or from an up state to a down state.

```
Device# configure terminal
Device(config)# snmp-server host 172.20.2.160 traps comaccess mpls-vpn
Device(config)# snmp-server enable traps mpls vpn vrf-up vrf-down
```

Example: Configuring Threshold Values for MPLS VPN SNMP Notifications

The following example shows how to set a maximum threshold or 10000 routes and a warning threshold that is 80 percent of the maximum threshold for a VRF named vpn1 on a device:

```
Device(config)# ip vrf vpn1
Device(config)# maximum routes 10000 80
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Cisco IOS Multiprotocol Label Switching Command Reference
MPLS Virtual Private Network (VPN) configuration tasks	"MPLS Virtual Private Networks" module in the *MPLS Layer 3 VPNs Configuration Guide*

MIBs

MIBs	MIBs Link
MPLS/BGP Virtual Private Network Management Information Base Using SMIv2 (*draft-ietf-ppvpn-mpls-vpn-mib-03.txt*) MPLS-VPN-MIB.my	To obtain lists of supported MIBs by platform and Cisco software release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL: http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 2233	*The Interfaces Group MIB using SMIv2*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for MPLS VPN SNMP Notifications

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 73: Feature Information for MPLS VPN SNMP Notifications

Feature Name	Releases	Feature Information
MPLS VPN SNMP Notifications	12.0(21)ST 12.0(22)S 12.2(13)T	The MPLS VPN SNMP Notifications feature provides Simple Network Management Protocol (SNMP) agent support in Cisco software for Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) event notifications. In Cisco IOS Release 12.0(21)ST, this feature was introduced. In Cisco IOS Release 12.0(22)S, this feature was integrated. In Cisco IOS Release 12.2(13)T, this feature was integrated. Supported platforms: • Cisco IOS 12.0 S and ST Releases: Cisco 7500 series, Cisco 12000 series. • Cisco IOS 12.2 T Releases: Cisco 3620, Cisco 3640, Cisco 7200 series, Cisco 7500 series, Cisco MGX 8850-RPM. **Note** In Cisco IOS Releases 12.0(21)ST and 12.0(22)S, the PPVPN MPLS-VPN-MIB notifications are described in the *MPLS VPN--SNMP MIB Support* feature module. The following commands were introduced or modified: **snmp-server enable traps mpls vpn**, **snmp-server host**.

Glossary

ASN.1—Abstract Syntax Notation One. OSI language for describing data types independent of particular computer structures and representation techniques. Described by ISO International Standard 8824.

BGP—Border Gateway Protocol. The exterior Border Gateway Protocol used to exchange routing information between routers in separate autonomous systems. BGP uses Transmission Control Protocol (TCP). Because TCP is a reliable protocol, BGP does not experience problems with dropped or fragmented data packets.

CEF—Cisco Express Forwarding. An advanced Layer 3 IP switching technology. CEF optimizes network performance and scalability for networks with large and dynamic traffic patterns.

CE device—customer edge router. A router on the border between a VPN provider and a VPN customer that belongs to the customer.

community—In SNMP, a logical group of managed devices and NMSs in the same administrative domain.

community name—*See* community string.

community string—Text string that acts as a password and is used to authenticate messages sent between a managed station and a router containing an SNMP agent. The community string is sent in every packet between the manager and the client. Also called a community name.

IETF—Internet Engineering Task Force. Task force consisting of over 80 working groups responsible for developing Internet standards. The IETF operates under the auspices of ISOC. *See also* ISOC.

informs—A type of notification message that is more reliable than a conventional trap notification message, because the informs message notification requires acknowledgment, and a trap notification does not.

ISOC—Internet Society. International nonprofit organization, founded in 1992, that coordinates the evolution and use of the Internet. In addition, ISOC delegates authority to other groups related to the Internet, such as the IAB. ISOC is headquartered in Reston, Virginia (United States).

label—A short, fixed-length data construct that tells switching nodes how to forward data (packets or cells).

label distribution protocol—*See* LDP.

label forwarding information base—*See* LFIB.

label switch router—*See* LSR.

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. In the Cisco Label Switching system, the data structure for storing information about incoming and outgoing tags (labels) and associated equivalent packets suitable for labeling.

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

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

MPLS —Multiprotocol Label Switching. 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.

MPLS interface—An interface on which MPLS traffic is enabled.

MPLS VPN—Multiprotocol Label Switching Virtual Private Network. Using MPLS VPNs in a Cisco network provide the capability to deploy and administer scalable Layer 3 VPN backbone services including applications, data hosting network commerce, and telephony services, to business customers. A VPN is a secure IP-based network that shares resources on one or more physical networks. A VPN contains geographically dispersed sites that can communicate securely over a shared backbone.

For an MPLS VPN Solution, an MPLS VPN is a set of PEs that are connected by means of a common "backbone" network to supply private IP interconnectivity between two or more customer sites for a given customer. Each VPN has a set of provisioning templates and policies and can span multiple provider administrative domains (PADs).

Multiprotocol label Switching—*See* MPLS.

notification —A message sent by an SNMP agent to a network management station, console, or terminal to indicate that a significant event within Cisco software has occurred. *See also* trap.

NMS —network management system. A powerful, well-equipped computer (typically an engineering workstation) that is used by a network administrator to communicate with other devices in the network. An NMS is typically used to manage network resources, gather statistics, and perform a variety of network administration and configuration tasks.

PE device—provider edge router. A router on the border between a VPN provider and a VPN customer that belongs to the provider.

PPVPN —Provider-Provisioned VPN. The name of the IETF working group that is developing the PPVPN-MPLS-VPN MIB (MPLS-VPN-MIB).

QoS —quality of service. Measure of performance for a transmission system that reflects its transmission quality and service availability.

RSVP —Resource Reservation Protocol. Protocol for reserving network resources to provide Quality of Service guarantees to application flows.

Simple Network Management Protocol—*See* SNMP.

SNMP —Simple Network Management Protocol. Network management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, and to manage configurations, statistics collection, performance, and security. See *also* SNMP2.

SNMP2 —SNMP Version 2. Version 2 of the popular network management protocol. SNMP2 supports centralized as well as distributed network management strategies, and includes improvements in the SMI, protocol operations, management architecture, and security. *See also* SNMP.

traffic engineering—The techniques and processes used to cause routed traffic to travel through the network on a path other than the one that would have been chosen if standard routing methods had been used.

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

VPN —Virtual Private Network. A group of sites that, as the result of a set of administrative policies, are able to communicate with each other over a shared backbone network. *See* MPLS VPN.

VRF —VPN routing/forwarding instance. A VRF consists of an IP routing table, a derived forwarding table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols that determine what goes into the forwarding table. In general, a VRF includes the routing information that defines a customer VPN site that is attached to a PE device.

INDEX