



Multiprotocol Label Switching on Cisco Routers

This document describes commands for configuring and monitoring MPLS functionality on Cisco routers and switches. It is intended to be used as a companion document to similar publications describing other MPLS applications (see the section entitled “Related Documents”).

This document includes the following sections:

- Supported Platforms
- Supported Standards, MIBs, and RFCs
- Functional Description of Multiprotocol Label Switching
- Prerequisites
- Configuration Tasks
- Saving Configurations: MPLS/Tag Switching Commands
- MPLS Command Summary
- Command Reference
- Debug Commands
- Glossary

Feature Overview

Multiprotocol label switching (MPLS) combines the performance and capabilities of Layer 2 (data link layer) switching with the proven scalability of Layer 3 (network layer) routing. MPLS enables service providers to meet the challenges of explosive growth in network utilization while providing the opportunity to differentiate services without sacrificing the existing network infrastructure. The MPLS architecture is flexible and can be employed in any combination of Layer 2 technologies. MPLS support is offered for all Layer 3 protocols, and scaling is possible well beyond that typically offered in today’s networks.

MPLS efficiently enables the delivery of IP services over an ATM switched network. MPLS supports the creation of different routes between a source and a destination on a purely router-based Internet backbone. By incorporating MPLS into their network architecture, service providers can save money, increase revenue and productivity, provide differentiated services, and gain competitive advantages.

MPLS Benefits

MPLS provides the following major benefits to service provider networks:

- Scalable support for virtual private networks (VPNs)—MPLS enables VPN services to be supported in service provider networks, thereby greatly accelerating Internet growth.

The use of MPLS for VPNs provides an attractive alternative to the building of VPNs by means of either ATM or Frame Relay permanent virtual circuits (PVCs) or various forms of tunneling to interconnect routers at customer sites.

Unlike the PVC VPN model, the MPLS VPN model is highly scalable and can accommodate increasing numbers of sites and customers. The MPLS VPN model also supports “any-to-any” communication among VPN sites without requiring a full mesh of PVCs or the backhauling (suboptimal routing) of traffic across the service provider network. For each MPLS VPN user, the service provider’s network appears to function as a private IP backbone over which the user can reach other sites within the VPN organization, but not the sites of any other VPN organization.

From a user perspective, the MPLS VPN model enables network routing to be dramatically simplified. For example, rather than having to manage routing over a topologically complex virtual backbone composed of many PVCs, an MPLS VPN user can generally employ the service provider’s backbone as the default route in communicating with all of the other VPN sites.

- Explicit routing capabilities (also called constraint-based routing or traffic engineering)—Explicit routing employs “constraint-based routing,” in which the path for a traffic flow is the shortest path that meets the resource requirements (constraints) of the traffic flow.

In MPLS traffic engineering, factors such as bandwidth requirements, media requirements, and the priority of one traffic flow versus another can be taken into account. These traffic engineering capabilities enable the administrator of a service provider network to

- Control traffic flow in the network
- Reduce congestion in the network
- Make best use of network resources

Thus, the network administrator can specify the amount of traffic expected to flow between various points in the network (thereby establishing a traffic matrix), while relying on the routing system to

- Calculate the best paths for network traffic
- Set up the explicit paths to carry the traffic

- Support for IP routing on ATM switches (also called IP and ATM integration)—MPLS enables an ATM switch to perform virtually all of the functions of an IP router. This capability of an ATM switch stems from the fact that the MPLS forwarding paradigm, namely, label swapping, is exactly the same as the forwarding paradigm provided by ATM switch hardware.

The key difference between a conventional ATM switch and an ATM label switch is the control software used by the latter to establish its virtual channel identifier (VCI) table entries. An ATM label switch uses IP routing protocols and the Tag Distribution Protocol (TDP) to establish VCI table entries.

An ATM label switch can function as a conventional ATM switch. In this dual mode, the ATM switch resources (such as VCI space and bandwidth) are partitioned between the MPLS control plane and the ATM control plane. The MPLS control plane provides IP-based services, while the ATM control plane supports ATM-oriented functions, such as circuit emulation or PVC services.

Restrictions

Label switching on a router requires that Cisco Express Forwarding (CEF) be enabled on that router. Refer to the Cisco Express Forwarding (CEF) feature documentation for configuration information.

Related Documents

For additional information about MPLS functionality running on routers or switches in a network, consult the following documentation for Cisco IOS Release 12.1(3)T:

- *MPLS Class of Service*—This feature enables network administrators to provide a range of differentiated services across an MPLS network. Such services are implemented by means of an appropriate setting of the IP precedence bit in each transmitted IP packet.
- *MPLS Traffic Engineering and Enhancements*—This feature enables an MPLS backbone to replicate and expand upon the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks. In service provider and Internet service provider (ISP) backbones, traffic engineering provides an effective means of managing networks. Such backbones must support high transmission capacities and be resilient to link or node failures.
- *MPLS Virtual Private Networks (VPNs)*—This feature enables users to deploy and administer IPv4 Layer 3, value-added services and business applications across a public network infrastructure. By deploying business applications on a broad scale over wide area networks (WANs), MPLS VPN users can reduce costs, increase revenue, and develop new business opportunities.

Supported Platforms

MPLS is supported on the following platforms:

- Cisco LightStream 1010 ATM switch—For information about label switching configuration and command syntax on the LightStream 1010 ATM switch, see the *LightStream 1010 ATM Switch Software Configuration Guide* Release 11.3.
- Cisco 2600 series routers
- Cisco RSP7000 route switch processor
- Cisco 7200 series routers
- Cisco 7500 series routers
- Cisco 12000 series GSR routers

Supported Standards, MIBs, and RFCs

The supported standards, MIBs, and RFCs applicable to the MPLS applications listed above under Related Documents appear in the respective feature module for the application.

Functional Description of Multiprotocol Label Switching

Label switching is a high-performance packet forwarding technology that integrates the performance and traffic management capabilities of data link layer (Layer 2) switching with the scalability, flexibility, and performance of network layer (Layer 3) routing.

Label Switching Functions

In conventional Layer 3 forwarding mechanisms, as a packet traverses the network, each router extracts all the information relevant to forwarding the packet from the Layer 3 header. This information is then used as an index for a routing table lookup to determine the next hop for the packet.

In the most common case, the only relevant field in the header is the destination address field, but in some cases, other header fields might also be relevant. As a result, the header analysis must be done independently at each router through which the packet passes. In addition, a complicated table lookup must also be done at each router.

In label switching, the analysis of the Layer 3 header is done only once. The Layer 3 header is then mapped into a fixed length, unstructured value called a *label*.

Many different headers can map to the same label, as long as those headers always result in the same choice of next hop. In effect, a label represents a *forwarding equivalence class*—that is, a set of packets which, however different they may be, are indistinguishable by the forwarding function.

The initial choice of a label need not be based exclusively on the contents of the Layer 3 packet header; for example, forwarding decisions at subsequent hops can also be based on routing policy.

Once a label is assigned, a short label header is added at the front of the Layer 3 packet. This header is carried across the network as part of the packet. At subsequent hops through each MPLS router in the network, labels are swapped and forwarding decisions are made by means of MPLS forwarding table lookup for the label carried in the packet header. Hence, the packet header does not need to be reevaluated during packet transit through the network. Because the label is of fixed length and unstructured, the MPLS forwarding table lookup process is both straightforward and fast.

Distribution of Label Bindings

Each label switching router (LSR) in the network makes an independent, local decision as to which label value to use to represent a forwarding equivalence class. This association is known as a label binding. Each LSR informs its neighbors of the label bindings it has made. This awareness of label bindings by neighboring routers is facilitated by the following protocols:

- Tag Distribution Protocol (TDP)—Used to support MPLS forwarding along normally routed paths
- Resource Reservation Protocol (RSVP)—Used to support MPLS traffic engineering
- Border Gateway Protocol (BGP)—Used to support MPLS virtual private networks (VPNs)

When a labeled packet is being sent from LSR A to the neighboring LSR B, the label value carried by the IP packet is the label value that LSR B assigned to represent the forwarding equivalence class of the packet. Thus, the label value changes as the IP packet traverses the network.

Label Switch Path (LSP) Tunnel Configuration

LSP tunnels are calculated at the *headend* (transmit end) router, based on the best fit between the required resources and the available resources for the flow (the constraint-based routing model). The Interior Gateway Protocol (IGP) automatically routes the traffic flows onto these LSP tunnels. Typically, a packet crossing the MPLS traffic engineering backbone travels on a single LSP tunnel that connects the ingress router to the egress router.

You create and maintain LSP tunnels by means of the command line interface (CLI). The CLI commands you use for creating and maintaining LSP tunnels are described in the “Command Reference” section below.

MPLS Class of Service

MPLS class of service (CoS) functionality enables network administrators to provide differentiated services across an MPLS network. A range of networking requirements can be satisfied by specifying the particular class of service for each packet by means of the precedence bit in each packet. You can differentiate MPLS CoS services by setting the IP precedence bit in each transmitted packet.

MPLS CoS provides the following differentiated services:

- Packet classification
- Congestion avoidance
- Congestion management

MPLS CoS enables you to duplicate Cisco IOS IP CoS (Layer 3) features as closely as possible in MPLS devices, including label edge routers (LERs), label switching routers (LSRs), and asynchronous transfer mode LSRs (ATM LSRs). MPLS CoS functions map nearly one-for-one to IP CoS functions on all types of interfaces.

MPLS Traffic Engineering

MPLS traffic engineering functionality enables an MPLS backbone to replicate and expand upon the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks. Traffic engineering is especially important for the management of complex, high-bandwidth service provider and Internet service provider (ISP) backbones.

In conventional Layer 3 routing, network topologies frequently provide multiple paths between two points. The normal routing procedure is to select a single path as the Layer 3 route between the two points, regardless of the load on the links that implement the path. As a consequence, some links might be congested while other links are under utilized.

With MPLS, however, traffic engineering features are integrated into Layer 3 services, thus optimizing the routing of IP traffic in high utilization, high transmission capacity network backbones. In such operating environments, MPLS traffic engineering provides the following benefits:

- Enhances standard Interior Gateway Protocols (IGPs), such as IS-IS and OSPF, giving you the ability to automatically map packets onto appropriate traffic flows and to transport packets efficiently by means of MPLS forwarding.
- Determines the best routes for traffic flows across a network, based on the resources required by the traffic flow versus the available resources within the network.

- Employs “constraint-based routing” in which the path chosen for a traffic flow is the shortest path that meets the resource requirements (that is, the constraints) of the flow. In MPLS traffic engineering, a given traffic flow has its own bandwidth requirements, media requirements, and transmission priority versus other traffic flows.
- Recovers dynamically from link or node failures that result from changes in network topology. In these instances, MPLS adapts to a new set of “constraints.”

In addition, with MPLS traffic engineering, you can override the routing protocols used by multiple routers, and you can direct selected traffic to flow over specified paths in the network, giving you the capability to

- Balance network loading
- Use network resources more effectively
- Provide differentiated levels of service

MPLS Virtual Private Networks

MPLS VPN functionality enables service providers to deploy scalable VPNs and build a networking foundation through which value-added services can be delivered to Internet users. Among such value-added services are the following:

- **Connectionless Services**—An advantage of MPLS VPNs is that the services provided thereby are connectionless. In contrast, current VPN solutions impose a connection-oriented, point-to-point overlay on the network. In a connectionless MPLS VPN environment, however, no prior action is required to establish communication between hosts. Furthermore, network complexity is reduced because you do not need traffic tunnels and encryption to ensure privacy of communications.
- **Centralized Services**—Implementing MPLS VPNs in Layer 3 enables delivery of services to a targeted group of users structured as a VPN. A VPN provides a way to flexibly deliver such value-added services as the following to targeted customers:
 - IP multicast
 - Quality of service (QoS)
 - Telephony support
 - Video conferencing
 - Web hosting
- **Network scalability**—MPLS VPNs use a peer model and Layer 3 connectionless architecture to provide scalable VPN solutions. The peer model requires a customer site to peer only with one provider edge (PE) router, as opposed to all other customer premises equipment (CPE) or customer edge (CE) routers that are members of the VPN. The MPLS VPN connectionless architecture enables the establishment of VPNs in Layer 3, thereby eliminating the need for tunnels or virtual circuits (VCs).
- **Network security**—MPLS VPNs offer the same level of security as connection-oriented VPNs. Packets from one VPN do not inadvertently go to another VPN. For example, with MPLS VPNs, security is provided at two levels:
 - At the edge of a provider network, ensuring that packets received from a customer are placed on the correct VPN.

- At the backbone, VPN traffic is kept separate. Hence, malicious spoofing (an attempt to gain access to a PE router) is nearly impossible because the packets received from customers are IP packets and must be received on a particular interface or subinterface to be uniquely identified with a VPN label.
- Integrated class of service (CoS) support—Integrated VPN CoS services provide such benefits as the following:
 - Predictable performance
 - Consistent policy implementation
 - Support for multiple levels of service
- Straightforward migration paths—MPLS VPNs can be built across multiple network architectures, including IP, ATM, Frame Relay, and hybrid networks. Thus, migration to a new network architecture is simplified because:
 - MPLS support on customer edge (CE) routers is not required
 - Modifications to the customer’s intranet are not required

Prerequisites

Label switching on a router requires that CEF be enabled on the router. Refer to the chapters on CEF in the following documents for CEF configuration information:

- *Cisco IOS Switching Services Command Reference*, Release 12.0
- *Cisco IOS Command Reference*, Release 12.0

Configuration Tasks

This section tells you how to configure a router for MPLS forwarding by enabling CEF on the router.

Configuration tasks for other MPLS applications for Cisco IOS Release 12.1(3)T are described in the feature module documentation for the application. The “Related Documents” section above lists each application and briefly describes its function in an MPLS operating environment.

Configuring a Router for MPLS Forwarding

MPLS forwarding on routers requires that CEF be enabled. To enable CEF on a router, issue the following commands:

```
Router# configure terminal  
Router(config)# ip cef [ distributed ]
```

**Note**

For best MPLS forwarding performance, use the **distributed** option on routers that support this option.

Verifying Configuration of MPLS Forwarding

To verify that CEF has been configured properly, issue the **show ip cef summary** command, which generates output similar to that shown below:

```
Router# sho ip cef summary
IP CEF with switching (Table Version 49), flags=0x0
 43 routes, 0 reresolve, 0 unresolved (0 old, 0 new)
 43 leaves, 49 nodes, 56756 bytes, 45 inserts, 2 invalidations
 2 load sharing elements, 672 bytes, 2 references
 1 CEF resets, 4 revisions of existing leaves
 4 in-place modifications
refcounts: 7241 leaf, 7218 node

Adjacency Table has 18 adjacencies
Router#
```

Saving Configurations: MPLS/Tag Switching Commands

The MPLS commands described in this document have been derived from equivalent tag switching commands. During the transition period from a tag switching environment to a standards-based MPLS environment, several configuration commands with both MPLS and tag switching forms are being supported. For example, the **mpls ip** command is equivalent to the **tag-switching ip** command.

Refer to Table 1 in the MPLS Command Summary section below for the correspondence between the MPLS commands described in this document and their earlier tag switching forms.

During the transition period from tag switching to MPLS, the tag switching form of configuration commands (that have both MPLS and tag switching forms) is written to saved configurations. Suppose, for example, that you configure MPLS hop-by-hop forwarding for a router POS interface by means of the following commands:

```
Router# configure terminal
Router(config)# interface POS3/0
Router(config-if)# mpls ip
```

In this example, the **mpls ip** command has a tag switching form. After you enter these commands and save this configuration or display the running configuration by means of the **show running** command, the configuration commands thus saved or displayed appear as shown below:

```
interface POS3/0
tag-switching ip
```

Writing the tag switching form of commands (that have both tag switching and MPLS forms) to the saved configuration enables you to

- Use a new router software image to modify and write configurations
- Later use configurations created by the new image with earlier software versions that do not support the MPLS forms of commands

For the above example, older software that supports tag switching commands, but not new MPLS commands, could successfully interpret the interface configuration.

MPLS Command Summary

Table 1 summarizes the general-purpose MPLS commands described in this document. For the most part, these MPLS commands have been derived from existing tag-switching commands, thus preserving the basic syntax of previous commands in implementing new MPLS functionality.

Table 1 Summary of MPLS Commands Described in this Document


| Command | Corresponding Tag Switching Command | Description |
|--|---|--|
| <code>interface atm</code> | <code>interface atm</code> | Enters interface configuration mode, specifies ATM as the interface type, and enables the creation of a subinterface on the ATM interface. |
| <code>mpls atm control-vc</code> | <code>tag-switching atm control-vc</code> | Configures the VPI and VCI to be used for the initial link to the label switching peer device. |
| <code>mpls atm vpi</code> | <code>tag-switching atm vpi</code> | Configures the range of values to be used in the VPI field for label VCs. |
| <code>mpls ip (global configuration)</code> | <code>tag-switching ip (global configuration)</code> | Enables MPLS forwarding of IPv4 packets along normally routed paths for the platform. |
| <code>mpls ip (interface configuration)</code> | <code>tag-switching ip (interface configuration)</code> | Enables MPLS forwarding of IPv4 packets along normally routed paths for a particular interface. |
| <code>mpls ip default-route</code> | <code>tag-switching ip default-route</code> | Enables the distribution of labels associated with the IP default route. |
| <code>mpls ip propagate-ttl</code> | <code>tag-switching ip propagate-ttl</code> | Sets the time-to-live (TTL) value when an IP packet is encapsulated in MPLS. |
| <code>mpls label range</code> | <code>tag-switching tag-range downstream</code> | Configures the range of local labels available for use on packet interfaces.  Note The syntax of this command differs slightly from its tag-switching counterpart. |
| <code>mpls mtu</code> | <code>tag-switching mtu</code> | Sets the per-interface maximum transmission unit (MTU) for labeled packets. |
| <code>show mpls forwarding-table</code> | <code>show tag-switching forwarding-table</code> | Displays the contents of the label forwarding information base (LFIB). |
| <code>show mpls interfaces</code> | <code>show tag-switching interfaces</code> | Displays information about one or more interfaces that have been configured for label switching. |
| <code>show mpls label range</code> | N/A | Displays the range of local labels available for use on packet interfaces. |
| <code>debug mpls adjacency</code> | <code>debug tag-switching adjacency</code> | Displays changes to label switching entries in the adjacency database. |
| <code>debug mpls events</code> | <code>debug tag-switching events</code> | Displays information about significant MPLS events. |
| <code>debug mpls lfib cef</code> | <code>debug tag-switching tfib cef</code> | Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed. |

Table 1 Summary of MPLS Commands Described in this Document (continued)

| Command | Corresponding Tag Switching Command | Description |
|-------------------------------|--|--|
| debug mpls lfib enc | debug tag-switching tfib enc | Prints detailed information about label encapsulations while label rewrites are created or updated and placed into the label forwarding information base (LFIB). |
| debug mpls lfib lsp | debug tag-switching tfib tsp | Prints detailed information about label rewrites being created and deleted as TSP tunnels are added or removed. |
| debug mpls lfib state | debug tag-switching tfib state | Traces what happens when label switching is enabled or disabled. |
| debug mpls lfib struct | debug tag-switching tfib struct | Traces the allocation and freeing of LFIB-related data structures, such as the LFIB itself, label-rewrites, and label-info data. |
| debug mpls packets | debug tag-switching packets | Displays labeled packets switched by the host router. |

Command Reference

This section describes the following general-purpose MPLS commands:

- **interface atm**
- **mpls atm control-vc**
- **mpls atm vpi**
- **mpls ip (global configuration)**
- **mpls ip (interface configuration)**
- **mpls ip default-route**
- **mpls ip propagate-ttl**
- **mpls label range**
- **mpls mtu**
- **show mpls forwarding-table**
- **show mpls interfaces**
- **show mpls label range**

interface atm

To enter interface configuration mode, specify ATM as the interface type, and create a subinterface on that interface type, use the **interface atm** global configuration command. The subinterface for the ATM interface is created the first time this command is issued with a specified subinterface number.

interface atm *interface.subinterface-number* [**mpls** | **tag-switching** | **point-to-point** | **multipoint**]

Syntax Description

| | |
|----------------------------|---|
| <i>interface</i> | Specifies a (physical) ATM interface (for example, 3/0). |
| <i>subinterface-number</i> | Specifies the subinterface number for the ATM interface. On Cisco 7500 series routers, subinterface numbers can range from 0 to 4294967285. |
| mpls | (Optional.) Specifies MPLS as the interface type for which a subinterface is to be created. |
| tag-switching | (Optional.) Specifies tag-switching as the interface type for which a subinterface is to be created. |
| point-to-point | (Optional.) Specifies point-to-point as the interface type for which a subinterface is to be created. |
| multipoint | (Optional.) Specifies multipoint as the interface type for which a subinterface is to be created. |

Defaults

This command has no default behavior or values.

Command Modes

Global configuration.

Command History

| Release | Modification |
|----------|---|
| 10.0 | This command was introduced. |
| 12.1(3)T | This command was modified to introduce new optional subinterface types. |

Usage Guidelines

The **interface atm** command enables you to define a subinterface for a specified type of ATM interface.

Examples

For physical ATM interface 3/0, the following command creates an ATM MPLS subinterface having subinterface number 1:

```
Router# interface atm 3/0.1 mpls
```

Related Commands

| Command | Description |
|-----------------------------|---|
| show mpls interfaces | Displays information about one or more MPLS interfaces that have been configured for label switching. |

mpls atm control-vc

To configure the VPI and VCI to be used for the initial link to the label switching peer device, use the **mpls atm control-vc** interface configuration command. The initial link is used to establish the TDP session and to carry non-IP traffic. To clear the interface configuration, use the **no** form of this command.

```
mpls atm control-vc vpi vci
```

```
no mpls atm control-vc vpi vci
```

| Syntax Description | |
|--------------------|-----------------------------|
| <i>vpi</i> | Virtual path identifier. |
| <i>vci</i> | Virtual channel identifier. |

Defaults If the subinterface has not changed to a VP tunnel, the default is 0/32. If the subinterface corresponds to VP tunnel VPI X, the default is X/32.

Command Modes Interface configuration

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines For a router interface (for example, an AIP), ATM label switching can be enabled only on a label-switch subinterface.



Note

The **mpls atm control-vc** and **mpls atm vpi** subinterface level configuration commands are available on any interface that can support ATM labeling.

On the Cisco LightStream 1010 ATM switch, a subinterface corresponds to a VP tunnel; thus, the entry in the VPI field of the control-vc must match the entry in the VPI field of the VP tunnel.

Examples The following commands create a label switching subinterface on a router and select VPI 1 and VCI 34 as the control VC:

```
Router(config)# interface atm4/0.1 mpls
Router(config-if)# mpls ip
Router(config-if)# mpls atm control-vc 1 34
```

| Related Commands | Command | Description |
|------------------|----------------------|---|
| | show mpls interfaces | Displays information about one or more interfaces for which label switching has been enabled. |

mpls atm vpi

To configure the range of values to be used in the VPI field for label VCs, use the **mpls atm vpi** interface configuration command. To clear the interface configuration, use the **no** form of this command.

```
mpls atm vpi vpi [- vpi]
```

```
no mpls atm vpi vpi [- vpi]
```

Syntax Description

| | |
|--------------|--|
| <i>vpi</i> | Virtual path identifier (low end of range). |
| - <i>vpi</i> | (Optional.) Virtual path identifier (high end of range). |

Defaults

The default is 1-1.

Command Modes

Interface configuration

Command History

| Release | Modification |
|----------|--|
| 11.1CT | This command was introduced. |
| 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines

To configure ATM label switching on a router interface (for example, an ATM interface processor), you must enable a label switching subinterface.



Note

The **mpls atm control-vc** and **mpls atm vpi** interface configuration commands are available on any interface that can support ATM labeling.

Use this command to select an alternate range of VPI values for ATM label assignment on this interface. The two ends of the link negotiate a range defined by the intersection (overlapping of labels in common) of the range configured at each end of the connection.

Examples

In the following example, a subinterface is created and a VPI range from 1 to 3 is selected:

```
Router(config)# interface atm4/0.1 mpls
Router(config-if)# mpls ip
Router(config-if)# mpls atm vpi 1-3
```

Related Commands

| Command | Description |
|----------------------------|--|
| mpls atm control-vc | Configures the VPI and VCI to be used for the initial link to the label switching peer device. |

mpls ip (global configuration)

To enable MPLS forwarding of IPv4 packets along normally routed paths for the platform, use the **mpls ip** global configuration command. Use the **no** form of the command to disable this feature.

mpls ip

no mpls ip

Syntax Description This command has no optional keywords or arguments.

Defaults Label switching of IPv4 packets along normally routed paths is enabled for the platform.

Command Modes Global configuration

| Command History | Release | Modification |
|-----------------|----------|------------------------------|
| | 12.1(3)T | This command was introduced. |

Usage Guidelines This command enables MPLS forwarding of IPv4 packets along normally routed paths (sometimes called dynamic label switching). For a given interface to perform dynamic label switching, this function must be enabled for the interface and the platform.

The **no** form of this command stops dynamic label switching for all platform interfaces, regardless of the interface configuration; it also stops distribution of labels for dynamic label switching. However, the **no** form of this command does not affect the sending of labeled packets through TSP tunnels.

For an LC-ATM interface, the **no** form of this command prevents the establishment of label VCs originating at, terminating at, or passing through the platform.

Examples In the following example, dynamic label switching is disabled for the platform, terminating all label distribution for the platform:

```
Router(config)# no mpls ip
```

| Related Commands | Command | Description |
|------------------|--|---|
| | mpls ip (interface configuration) | Enables label switching of IPv4 packets along normally routed paths for the associated interface. |

mpls ip (interface configuration)

To enable MPLS forwarding of IPv4 packets along normally routed paths for a particular interface, use the **mpls ip** interface configuration command.

Use the **no** form of the command to disable this feature.

mpls ip

no mpls ip

Syntax Description

This command has no optional keywords or arguments.

Defaults

MPLS forwarding of IPv4 packets along normally routed paths for the interface is disabled.

Command Modes

Interface configuration

Command History

| Release | Modification |
|----------|------------------------------|
| 12.1(3)T | This command was introduced. |

Usage Guidelines

MPLS forwarding of IPv4 packets along normally routed paths is sometimes called dynamic label switching. If dynamic label switching has been enabled for the platform when this command is issued on an interface, you can start label distribution for the interface by initiating periodic transmission of neighbor discovery hello messages on the interface. When the outgoing label for a destination routed through the interface is known, packets for the destination are labeled with that outgoing label and forwarded through the interface.

The **no** form of this command causes packets routed out through the interface to be sent unlabeled; it also ends label distribution for the interface. The **no** form of this command does not affect the sending of labeled packets through any TSP tunnels that might use the interface.

For an LC-ATM interface, the **no** form of this command prevents the establishment of label VCs beginning at, terminating at, or passing through the interface.

Examples

In the following example, label switching is enabled on the Ethernet interface specified:

```
Router(config)# configure terminal
Router(config-if)# interface e0/2
Router(config-if)# mpls ip
```

Related Commands

| Command | Description |
|-----------------------------|--|
| show mpls interfaces | Displays information about one or more interfaces that have been configured for label switching. |

mpls ip default-route

To enable the distribution of labels associated with the IP default route, use the **mpls ip default-route** global configuration command.

mpls ip default-route

Syntax Description This command has no optional keywords or arguments.

Defaults No distribution of labels for the IP default route.

Command Modes Global configuration

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines Dynamic label switching (that is, distribution of labels based on routing protocols) must be enabled before you can use the **mpls ip default-route** command.

Examples The following commands enable the distribution of labels associated with the IP default route:

```
Router# configure terminal
Router(config)# mpls ip
Router(config)# mpls ip default-route
```

| Related Commands | Command | Description |
|------------------|--|---|
| | mpls ip (global configuration) | Enables MPLS forwarding of IPv4 packets along normally routed paths for the platform. |
| | mpls ip (interface configuration) | Enables MPLS forwarding of IPv4 packets along normally routed paths for a particular interface. |

mpls ip propagate-ttl

To set the time-to-live (TTL) value on output when IP packets are being encapsulated in MPLS, use the **mpls ip propagate-ttl** privileged EXEC command. Use the **no** form of the command to disable this feature.

mpls ip propagate-ttl

no mpls ip propagate-ttl

Syntax Description

This command has no optional keywords or arguments.

Defaults

The MPLS TTL value on packet output is set based on the IP TTL value on packet input.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|------------------------------|
| 12.1(3)T | This command was introduced. |

Usage Guidelines

The **mpls ip propagate-ttl** command causes a **traceroute** command to show all the hops traversed by the MPLS packet in the network.

The **no** form of the **mpls ip propagate-ttl** command causes a **traceroute** command to ignore all hops traversed by the MPLS packet in the network.

Examples

The following is an example of the **mpls ip propagate-ttl** command:

```
Router# mpls ip propagate-ttl
```

This command generates no output.

Related Commands

| Command | Description |
|-------------------|--|
| traceroute | Discovers the routes that packets follow in traveling through a network to their destinations. |

mpls label range

To configure the range of local labels available for use on packet interfaces, use the **mpls label range** global configuration command. Use the **no** form of this command to revert to the platform defaults.

mpls label range *min max*

no mpls label range

Syntax Description

| | |
|------------|---|
| <i>min</i> | The smallest label allowed in the label space. The default is 16. |
| <i>max</i> | The largest label allowed in the label space. The default is 1048575. |

Defaults

The default values for the arguments of this command are:

- *min*—16
- *max*—1048575

The labels 0 through 15 are reserved by the IETF (see draft-ietf-mpls-label-encaps-07.txt for details) and cannot be included in the range specified by the **mpls label range** command.

Command Modes

Global configuration

Command History

| Release | Modification |
|----------|--|
| 11.1CT | This command was introduced. |
| 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines

The label range defined by the **mpls label range** command is used by all MPLS applications that allocate local labels (for dynamic label switching, MPLS traffic engineering, MPLS VPNs, and so on).

If you specify a new label range that does not overlap the range currently in use, the new range will not take effect until the router is reloaded again.

Examples

In the following example, you are shown how to configure the size of the local label space. In this example, *min* is set with the value of 200, and *max* is set with the value of 120000. Since the new range does not overlap the current label range (assumed to be the default, that is, *min* 16 and *max* 100000), the new range will not take effect until the router is reloaded.

```
Router# configure terminal
Router(config)# mpls label range 200 120000
% Label range changes will take effect at the next reload.
Router(config)#
```

If you had specified a new range that overlaps the current range (for example, new range of *min* 16 and *max* 120000), then the new range would take effect immediately.

| Related Commands | Command | Description |
|-------------------------|-----------------------|---|
| | show mpls label range | Displays the range of the MPLS local label space. |

mpls mtu

To set the per-interface maximum transmission unit (MTU) for labeled packets, use the **mpls mtu** interface configuration command.

mpls mtu *bytes*

no mpls mtu

| Syntax Description | <i>bytes</i> | The MTU value in bytes. The minimum allowable value is 64; the maximum allowable value is interface dependent. |
|--------------------|--------------|--|
|--------------------|--------------|--|

Defaults Use the interface MTU if an MPLS MTU has not been configured.

Command Modes Interface configuration

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines If a labeled IPv4 packet exceeds the MPLS MTU size for the interface, Cisco IOS software fragments the packet. If a labeled non-IPv4 packet exceeds the MPLS MTU size, the packet is dropped.

All devices on a physical medium must have the same MPLS MTU value in order for MPLS to interoperate.

The MTU for labeled packets for an interface is determined as follows:

- If the **mpls mtu** *bytes* command has been used to configure an MPLS MTU, the MTU for labeled packets is *bytes*.
- Otherwise, if the **mpls mtu** *bytes* command has been used to configure an interface MTU, the MTU for labeled packets is *bytes*.
- Otherwise, the MTU for labeled packets is the default MTU for the interface.

Because labeling a packet makes it larger due to the label stack, it may be desirable for the MPLS MTU to be larger than the interface MTU or IP MTU in order to prevent the fragmentation of labeled packets, which would not be fragmented if they were unlabeled.



Note

Changing the interface MTU by means of the **mpls mtu** *bytes* command changes the MPLS MTU also. However, the **mpls mtu** *bytes* command does not change the interface MTU.

Examples

In the following example, the maximum labeled packet size for serial interface Serial0 is set to 3500 bytes:

```
Router(config)# interface serial10  
Router(config-if)# mpls mtu 3500
```

show mpls forwarding-table

To display the contents of the MPLS Forwarding Information Base (LFIB), use the **show mpls forwarding-table** user EXEC command.

```
show mpls forwarding-table [{network {mask | length} | labels label [- label] | interface interface
| next-hop address | lsp-tunnel [tunnel-id ]}] [detail]
```

| Syntax Description | | |
|------------------------------------|-------------|--|
| <i>network</i> | (Optional.) | Destination network number. |
| <i>mask</i> | | IP address of destination mask whose entry is to be shown. |
| <i>length</i> | | Number of bits in mask of destination. |
| labels <i>label - label</i> | (Optional.) | Shows only entries with specified local labels. |
| interface <i>interface</i> | (Optional.) | Shows only entries with specified outgoing interface. |
| next-hop <i>address</i> | (Optional.) | Shows only entries with specified neighbor as next hop. |
| lsp-tunnel <i>tunnel-id</i> | (Optional.) | Shows only entries with specified LSP tunnel, or all LSP tunnel entries. |
| detail | (Optional.) | Displays information in long form (includes length of encapsulation, length of MAC string, maximum transmission unit (MTU), and all labels). |

| Command Modes | |
|---------------|-----------|
| | User EXEC |

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

| Usage Guidelines | |
|------------------|---|
| | The optional parameters described above allow specification of a subset of the entire LFIB. |

Examples

The following shows sample output from the **show mpls forwarding-table** command:

```
Router# show mpls forwarding-table
```

| Local tag | Outgoing tag or VC | Prefix or Tunnel Id | Bytes switched | tag | Outgoing interface | Next Hop |
|-----------|--------------------|---------------------|----------------|-----|--------------------|-------------|
| 26 | Untagged | 10.253.0.0/16 | 0 | | Et4/0/0 | 172.27.32.4 |
| 28 | 1/33 | 10.15.0.0/16 | 0 | | AT0/0.1 | point2point |
| 29 | Pop tag | 10.91.0.0/16 | 0 | | Hs5/0 | point2point |
| | 1/36 | 10.91.0.0/16 | 0 | | AT0/0.1 | point2point |
| 30 | 32 | 10.250.0.97/32 | 0 | | Et4/0/2 | 10.92.0.7 |
| | 32 | 10.250.0.97/32 | 0 | | Hs5/0 | point2point |
| 34 | 26 | 10.77.0.0/24 | 0 | | Et4/0/2 | 10.92.0.7 |
| | 26 | 10.77.0.0/24 | 0 | | Hs5/0 | point2point |
| 35 | Untagged [T] | 10.100.100.101/32 | 0 | | Tu301 | point2point |
| 36 | Pop tag | 168.1.0.0/16 | 0 | | Hs5/0 | point2point |
| | 1/37 | 168.1.0.0/16 | 0 | | AT0/0.1 | point2point |

[T] Forwarding through a TSP tunnel.
View additional tagging info with the 'detail' option

The following shows sample output from the **show mpls forwarding-table** command when you specify the **detail** keyword:

```
Router# show mpls forwarding-table detail
```

| Local tag | Outgoing tag or VC | Prefix or Tunnel Id | Bytes switched | tag | Outgoing interface | Next Hop |
|-----------|--------------------|--|----------------|-----|--------------------|-------------|
| 26 | Untagged | 10.253.0.0/16 | 0 | | Et4/0/0 | 172.27.32.4 |
| | | MAC/Encaps=0/0, MTU=1504, Tag Stack{} | | | | |
| 28 | 1/33 | 10.15.0.0/16 | 0 | | AT0/0.1 | point2point |
| | | MAC/Encaps=4/8, MTU=4470, Tag Stack{1/33(vcd=2)} 00020900 00002000 | | | | |
| 29 | Pop tag | 10.91.0.0/16 | 0 | | Hs5/0 | point2point |
| | | MAC/Encaps=4/4, MTU=4474, Tag Stack{} | | | | |
| | | FF030081 | | | | |
| | 1/36 | 10.91.0.0/16 | 0 | | AT0/0.1 | point2point |
| | | MAC/Encaps=4/8, MTU=4470, Tag Stack{1/36(vcd=3)} 00030900 00003000 | | | | |
| 30 | 32 | 10.250.0.97/32 | 0 | | Et4/0/2 | 10.92.0.7 |
| | | MAC/Encaps=14/18, MTU=1500, Tag Stack{32} 006009859F2A00E0F7E984828847 00020000 | | | | |
| | 32 | 10.250.0.97/32 | 0 | | Hs5/0 | point2point |
| | | MAC/Encaps=4/8, MTU=4470, Tag Stack{32} FF030081 00020000 | | | | |
| 34 | 26 | 10.77.0.0/24 | 0 | | Et4/0/2 | 10.92.0.7 |
| | | MAC/Encaps=14/18, MTU=1500, Tag Stack{26} 006009859F2A00E0F7E984828847 0001A000 | | | | |
| | 26 | 10.77.0.0/24 | 0 | | Hs5/0 | point2point |
| | | MAC/Encaps=4/8, MTU=4470, Tag Stack{26} FF030081 0001A000 | | | | |
| 35 | Untagged | 10.100.100.101/32 | 0 | | Tu301 | point2point |
| | | MAC/Encaps=0/0, MTU=1504, Tag Stack{} | | | via Et4/0/2 | |
| 36 | Pop tag | 168.1.0.0/16 | 0 | | Hs5/0 | point2point |
| | | MAC/Encaps=4/4, MTU=4474, Tag Stack{} | | | | |
| | | FF030081 | | | | |
| | 1/37 | 168.1.0.0/16 | 0 | | AT0/0.1 | point2point |
| | | MAC/Encaps=4/8, MTU=4470, Tag Stack{1/37(vcd=4)} 00040900 00004000 | | | | |

Table 2 describes the significant fields in this display.

Table 2 *show mpls forwarding-table Command Field Descriptions*

| Field | Description |
|---------------------|---|
| Local tag | Label assigned by this router. |
| Outgoing tag or VC | Label assigned by next hop, or VPI/VCI used to get to next hop. Some of the entries that you can specify in this column are: [T]—Means forwarding through a TSP tunnel. “Untagged”—Means there is no label for the destination from the next hop, or label switching is not enabled on the outgoing interface. “Pop tag”—Means that the next hop advertised an implicit NULL label for the destination, and that this router popped the top label. |
| Prefix or Tunnel Id | Address or tunnel to which packets with this label are going. |
| Bytes tag switched | Number of bytes switched with this incoming label. |
| Outgoing interface | Interface through which packets with this label are sent. |
| Next Hop | IP address of neighbor that assigned the outgoing label. |
| Mac/Encaps | Length in bytes of Layer 2 header, and length in bytes of packet encapsulation, including Layer 2 header and label header. |
| MTU | Maximum transmission unit (MTU) of labeled packet. |
| Tag Stack | All the outgoing labels. If the outgoing interface is TC-ATM, the VCD is also shown. |
| 00020900 00002000 | The actual encapsulation in hexadecimal form. There is a space shown between Layer 2 and the label header. |

show mpls interfaces

To display information about one or more interfaces that have been configured for label switching, use the **show mpls interfaces** user EXEC command.

show mpls interfaces [*interface*] [**detail**] [**all**]

| Syntax Description | | |
|--------------------|------------------|--|
| | <i>interface</i> | (Optional.) The interface about which to display label switching information. |
| | detail | (Optional.) Displays detailed label switching information by interface. |
| | all | (Optional.) Displays all the device interfaces that have MPLS applications associated with the interfaces. |

Defaults If no optional parameter or keyword is specified, summary information is displayed for each interface that has been configured for label switching.

Command Modes User EXEC

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines This command shows MPLS information about the specified interface, or about all of the interfaces for which MPLS has been configured.

Examples The following is sample output generated by the **show mpls interfaces** command:

```
Router# show mpls interfaces
Interface          IP          Tunnel    Operational
Ethernet1/1/1     Yes (tdp)  No       No
Ethernet1/1/2     Yes (tdp)  Yes      No
Ethernet1/1/3     Yes (tdp)  Yes      Yes
POS2/0/0          Yes (tdp)  No       No
ATM0/0.1          Yes (tdp)  No       No          (ATM labels)
ATM3/0.1          Yes (ldp)  No       Yes          (ATM labels)
ATM0/0.2          Yes (tdp)  No       Yes
```



Note

If an interface uses LC-ATM procedures, the associated line in the display is flagged with the following notation “(ATM labels)”.

Table 3 describes the significant fields in the sample display shown above.

Table 3 *show mpls interfaces Command Field Descriptions*

| Field | Description |
|-------------|---|
| Interface | Interface name. |
| IP | “Yes” if IP label switching (sometimes called hop-by-hop label switching) has been enabled on this interface. |
| Tunnel | “Yes” if LSP tunnel labeling has been enabled on this interface. |
| Operational | Operational state. “Yes” if packets are being labeled. |
| MTU | Maximum number of data bytes per labeled packet that will be transmitted. |

The following is sample output from the **show mpls interfaces** command when you specify the **detail** keyword:

```
Router# show mpls interfaces detail
Interface Ethernet1/1/1:
    IP labeling enabled (tdp)
    LSP Tunnel labeling not enabled
    MPLS operational
    MPLS turbo vector
    MTU = 1500
Interface POS2/0/0:
    IP labeling enabled (ldp)
    LSP Tunnel labeling not enabled
    MPLS not operational
    MPLS turbo vector
    MTU = 4470
Interface ATM3/0.1:
    IP labeling enabled (ldp)
    LSP Tunnel labeling not enabled
    MPLS operational
    MPLS turbo vector
    MTU = 4470
    ATM labels: Label VPI = 1
                 Label VCI range = 33 - 65535
                 Control VC = 0/32
```

Related Commands

| Command | Description |
|---|--|
| mpls ip (global configuration) | Enables label switching of IPv4 packets on all interfaces. |
| mpls ip (interface configuration) | Enables label switching of IPv4 packets on the associated interface. |
| mpls traffic-eng tunnels (global configuration) | Enables MPLS traffic engineering tunnel signaling on a device. This command is described in the document entitled <i>MPLS Traffic Engineering Feature Module</i> . |
| mpls traffic-eng tunnels (interface configuration) | Enables MPLS traffic engineering tunnel signaling on an interface. This command is described in the document entitled <i>MPLS Traffic Engineering Feature Module</i> . |

show mpls label range

To display the range of local labels available for use on packet interfaces, use the **show mpls label range** privileged EXEC command.

show mpls label range

Syntax Description This command has no optional keywords or arguments

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|------------------------------|
| | 12.0(9)ST | This command was introduced. |

Usage Guidelines You can use the **mpls label range** command to configure a range for local labels that is different from the default range. If the newly configured range does not overlap the current range, then the new range will not take effect until the router is reloaded. In this situation, the **show mpls label range** command displays both the label range currently in use and the label range that will be in use following the next router reload.

Examples In the following example, the use of the **show mpls label range** command is shown before and after the **mpls label range** command is used to configure a label range that does not overlap the starting label range.

```
Router# show mpls label range
Downstream label pool: Min/Max label: 16/100000
Router#

Router# configure terminal
Router(config)# mpls label range 200 120000
% Label range changes will take effect at the next reload.
Router(config)# exit

Router# show mpls label range
Downstream label pool: Min/Max label: 16/100000
    [Configured range for next reload: Min/Max label: 200/120000]
Router#
```

| Related Commands | Command | Description |
|------------------|-------------------------|---|
| | mpls label range | Configures range of values for use as local labels. |

Debug Commands

This section describes the following general-purpose MPLS debug commands:

- **debug mpls adjacency**
- **debug mpls events**
- **debug mpls lfib cef**
- **debug mpls lfib enc**
- **debug mpls lfib lsp**
- **debug mpls lfib state**
- **debug mpls lfib struct**
- **debug mpls packets**



Note

The output generated by the debug commands described below uses the older tag switching terminology in many places, rather than the newer MPLS IETF terminology. Over time, the output generated by these commands will be updated to reflect the new MPLS IETF terminology.

debug mpls adjacency

To display changes to label switching entries in the adjacency database, use the **debug mpls adjacency EXEC** command. The **no** form of this command disables debugging output.

debug mpls adjacency

no debug mpls adjacency

Usage Guidelines

This command has no optional keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|--|
| 11.1CT | This command was introduced. |
| 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines

Use the **debug mpls adjacency** command to monitor when entries are updated in or added to the adjacency database.

Examples

The following is sample output generated by the **debug mpls adjacency** command:

```
Router# debug mpls adjacency
TAG ADJ: add 10.10.0.1, Ethernet0/0/0
TAG ADJ: update 10.10.0.1, Ethernet0/0/0
```

Table 4 describes the significant fields shown in the sample display above.

Table 4 debug mpls adjacency Command Field Description

| Field | Description |
|---------------|---|
| add | Adding an entry to the database. |
| update | Updating the MAC address for an existing entry. |
| 10.10.0.1 | Address of neighbor TSR. |
| Ethernet0/0/0 | Connecting interface. |

debug mpls events

To display information about significant MPLS events, use the **debug mpls events** privileged EXEC command. Use the **no** form of this command to disable this feature.

debug mpls events

no debug mpls events

Syntax Description This command has no optional keywords or arguments.

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|----------|------------------------------|
| | 12.1(3)T | This command was introduced. |

Usage Guidelines Use this command to monitor significant MPLS events. For this IOS release, the only events reported by this command are changes to the MPLS router ID.

Examples The following is sample output from the **debug mpls events** command:

```
Router# debug mpls events
MPLS events debugging is on
```

```
TAGSW: Unbound IP address, 155.0.0.55, from Router ID
TAGSW: Bound IP address, 199.44.44.55, to Router ID
```

debug mpls lfib cef

To print detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed, use the **debug mpls lfib cef** EXEC command. The **no** form of this command disables debugging.

debug mpls lfib cef

no debug mpls lfib cef

Syntax Description This command has no keywords or arguments.

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|------------------------|----------------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines Several lines of output are produced for each route placed into the LFIB. If your router has thousands of labeled routes, be careful about issuing this command. When label switching is first enabled, each of these routes is placed into the LFIB, and several lines of output are displayed for each route.

Examples

The following is sample output displayed when you enter the **debug mpls lfib cef** command:

```
Router# debug mpls lfib cef

Cisco Express Forwarding related TFIB services debugging is on

tagcon: tc_ip_rtlookup fail on 10.0.0.0/8:subnet_lookup failed
TFIB: route tag chg 10.7.0.7/32,idx=1,inc=Withdrn,outg=Withdrn,enabled=0x2
TFIB: fib complete delete: prefix=10.7.0.7/32,inc tag=26,delete_info=1
TFIB: deactivate tag rew for 10.7.0.7/32,index=0
TFIB: set fib rew: pfx 10.7.0.7/32,index=0,add=0,tag_rew->adj=Ethernet2/3
TFIB: resolve tag rew,prefix=10.7.0.7/32,no tag_info,no parent
TFIB: fib scanner start:needed:1,unres:0,mac:0,loadinfo:0
TFIB: resolve tag rew,prefix=10.7.0.7/32,no tag_info,no parent
TFIB: fib upd loadinf 10.100.100.100/32,tag=Tun_hd,fib no loadin,tfib no loadin
TFIB: fib check cleanup for 10.100.100.100/32,index=0,return_value=0
TFIB: fib_scanner_end
TFIB: create dynamic entry for 10.11.0.11/32
TFIB: call find_route_tags,dist_method=1,next_hop=10.93.0.11,Et2/3
TFIB: route tag chg 10.11.0.11/32,idx=0,inc=26,outg=Unkn,enabled=0x3
TFIB: create tag info 10.11.0.11/32,inc tag=26,has no info
TFIB: resolve tag rew,prefix=10.11.0.11/32,has tag_info,no parent
TFIB: finish fib res 10.11.0.11/32:index 0,parent outg tag no parent
TFIB: fib upd loadinf 10.11.0.11/32,tag=26,fib no loadin,tfib no loadin
TFIB: set fib rew: pfx 10.11.0.11/32,index=0,add=1,tag_rew->adj=Ethernet2/3
tagcon: route_tag_change for: 10.250.0.97/32
      intag 33, outtag 28, nexthop tsr 10.11.0.11:0
TFIB: route tag chg 10.250.0.97/32,idx=0,inc=33,outg=28,enabled=0x3
TFIB: deactivate tag rew for 10.250.0.97/32,index=0
TFIB: set fib rew: pfx 10.250.0.97/32,index=0,add=0,tag_rew->adj=Ethernet2/3
TFIB: create tag info 10.250.0.97/32,inc tag=33,has old info
On VIP:
TFIB: route tag chg 10.13.72.13/32,idx=0,inc=34,outg=Withdrn,enabled=0x3
TFIB: deactivate tag rew for 10.13.72.13/32,index=0
TFIB: set fib rew: pfx 10.13.72.13/32,index=0,add=0,tag_rew->adj=
TFIB: create tag info 10.13.72.13/32,inc tag=34,has old info
TFIB: resolve tag rew,prefix=10.13.72.13/32,has tag_info,no parent
TFIB: finish fib res 10.13.72.13/32:index 0,parent outg tag no parent
TFIB: set fib rew: pfx 10.100.100.100/32,index=0,add=0,tag_rew->adj=
TFIB: create tag info 10.100.100.100/32,inc tag=37,has old info
TFIB: resolve tag rew,prefix=10.100.100.100/32,has tag_info,no parent
TFIB: finish fib res 10.100.100.100/32:index 0,parent outg tag no parent
TFIB: fib upd loadinf 10.100.100.100/32,tag=37,fib no loadin,tfib no loadin
```

Table 5 lists the significant fields in the sample display above.

See Table 7 for a description of special labels that appear in the output of this debug command.

Table 5 debug mpls lfib cef Command Field Descriptions

| Field | Description |
|--|--|
| tagcon | The name of the subsystem issuing the debug output (Label Control). |
| LFIB | The name of the subsystem issuing the debug output. |
| tc_ip_rtlookup fail on x.y.w.z/m: subnet_lookup failed | The destination with IP address and mask shown is not in the routing table. |
| route tag chg x.y.w.z/m | Request to create the LFIB entry for the specified prefix/mask. |
| idx=-1 | The index within the FIB entry of the path whose LFIB entry is being created. The parameter -1 means all paths for this FIB entry. |

Table 5 debug mpls lfib cef Command Field Descriptions (continued)

| Field | Description |
|---------------------------------------|--|
| inc=s | Incoming label of the entry being processed. |
| outg=s | Outgoing label of the entry being processed. |
| enabled=0xn | Bit mask indicating the types of label switching currently enabled: 0x1 = dynamic 0x2 = TSP tunnels 0x3 = both |
| fib complete delete | Indicates that the FIB entry is being deleted. |
| prefix=x.y.w.z/m | A destination prefix. |
| delete_info=1 | Indicates that label_info is also being deleted. |
| deactivate tag rew for x.y.w.z/m | Indicates that label rewrite for specified prefix is being deleted. |
| index=n | Index of path in the FIB entry being processed. |
| set fib rew: pfx x.y.w.z/m | Indicates that label rewrite is being installed or deleted from the FIB entry for the specified destination for label imposition purposes. |
| add=0 | Indicates that label rewrite is being deleted from the FIB (no longer imposing labels). |
| tag_rew->adj=s | Adjacency of label_rewrite for label imposition. |
| resolve tag rew,prefix=x.y.w.z/m | Indicates that the FIB route to the specified prefix is being resolved. |
| no tag_info | Indicates that there is no label_info for the destination (destination not labeled). |
| no parent | Indicates that route is not recursive. |
| fib scanner start | Indicates that the periodic scan of the FIB has started. |
| needed:1 | Indicates that LFIB needs the FIB to be scanned. |
| unres:n | Indicates the number of unresolved TFIB entries. |
| mac:n | Indicates the number of TFIB entries missing MAC strings. |
| loadinfo:n | Indicates whether the nonrecursive accounting state has changed and whether the loadinfo information in the LFIB needs to be adjusted. |
| fib upd loadinf x.y.w.z/m | Indicates that a check for nonrecursive accounting is being made and that the LFIB loadinfo information for the specified prefix is being updated. |
| tag=s | Incoming label of entry. |
| fib no loadin | Indicates that the corresponding FIB entry has no loadinfo. |
| tfib no loadin | Indicates that the LFIB entry has no loadinfo. |
| fib check cleanup for x.y.w.z/m | Indicates that a check is being made on the LFIB entry for the specified destination to determine if rewrite needs to be removed from the LFIB. |
| return_value=x | If x is 0, indicates that no change has occurred in the LFIB entry. If x is 1, there was a change. |
| fib_scanner_end | Indicates that the FIB scan has come to an end. |
| create dynamic entry for x.y.w.z/m | Indicates that the LFIB has been enabled and that an LFIB entry is being created for the specified destination. |
| call find_route_tags | Indicates that the labels for that destination are being requested. |
| dist_method=n | Identifies the label distribution method—TDP, TC-ATM, and so on. |
| next_hop=x.y.z.w | Identifies the next hop for the destination. |
| interface name | Identifies the outgoing interface for the destination. |

Table 5 debug mpls lfib cef Command Field Descriptions (continued)

| Field | Description |
|---|--|
| create tag info | Indicates that a label_info data structure is being created for the destination. |
| has no info | Indicates that the destination does not already have label_info. |
| finish fib re x.y.z.w/m | Indicates that the LFIB entry for the specified route is being completed. |
| parent outg tag s | If recursive, specifies the outgoing label of the route through which it is recursive (the parent). If not recursive, s = "no parent." |
| tagcon: route_tag_change for: x.y.z.w/m | Indicates that label control is notifying LFIB that labels are available for the specified destination. |
| intag s | Identifies the incoming label for the destination. |
| outtag s | Identifies the outgoing label for the destination. |
| nexthop tsr x.y.z.w.i | Identifies the TDP ID of the next hop which sent the tag. |

Related Commands

| Command | Description |
|-------------------------------|---|
| debug mpls lfib cef | Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed. |
| debug mpls lfib lsp | Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed. |
| debug mpls lfib state | Traces what happens when label switching is enabled or disabled. |
| debug mpls lfib struct | Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label-rewrites, and label-info data. |

debug mpls lfib enc

To print detailed information about label encapsulations while label rewrites are created or updated and placed in the label forwarding information base (LFIB), use the **debug mpls lfib enc** privileged EXEC command. The command output shows you which adjacency the label rewrite is being created on and the labels assigned. The **no** form of this command disables debugging output.

debug mpls lfib enc

no debug mpls lfib enc

Syntax Description This command has no keywords or arguments.

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines Several lines of output are produced for each route placed into the LFIB. If your router has thousands of labeled routes, issue this command with care. When label switching is first enabled, each of these routes is placed into the LFIB and a label encapsulation is created.

Examples The following is an example of output generated when you issue the **debug mpls lfib enc** command. This example shows the encapsulations for three routes that have been created and placed into the LFIB.

```
Router# debug mpls lfib enc

TFIB: finish res:inc tag=28,outg=Imp_null,next_hop=10.93.72.13,Ethernet4/0/3
TFIB: update_mac, mac_length = 14,addr=10.93.72.13,idb=Ethernet4/0/3
TFIB: get ip adj: addr=10.93.72.13,is_p2p=0, fibidb=Ethernet4/0/3,linktype=7
TFIB: get tag adj: addr=10.93.72.13,is_p2p=0, fibidb=Ethernet4/0/3,linktype=79
TFIB: encaps:inc=28,outg=Imp_null,idb:Ethernet4/0/3,sizes 14,14,1504,type 0
TFIB: finish res:inc tag=30,outg=27,next_hop=10.93.72.13,Ethernet4/0/3
TFIB: get ip adj: addr=10.93.72.13,is_p2p=0, fibidb=Ethernet4/0/3,linktype=7
TFIB: get tag adj: addr=10.93.72.13,is_p2p=0, fibidb=Ethernet4/0/3,linktype=79
TFIB: encaps:inc=30,outg=27,idb:Ethernet4/0/3,sizes 14,18,1500,type 0
TFIB: finish res:inc tag=30,outg=10,next_hop=0.0.0.0,ATM0/0.1
TFIB: get ip adj: addr=0.0.0.0,is_p2p=1, fibidb=ATM0/0.1,linktype=7
TFIB: get tag adj: addr=0.0.0.0,is_p2p=1, fibidb=ATM0/0.1,linktype=79
TFIB: encaps:inc=30,outg=10,idb:ATM0/0,sizes 4,8,4470,type 1
```

Table 6 describes the significant fields in the **debug mpls lfib enc** command output shown above.

Table 6 debug mpls lfib enc Command Field Descriptions

| Field | Description |
|--------------------|--|
| TFIB | Identifies the source of the message as the LFIB subsystem. |
| finish res | Shows that the LFIB resolution is being finished. |
| inc tag=x or inc=x | An incoming (local) label for the LFIB entry is being created. Labels can be numbers or special values. |
| outg=y | An outgoing (remote) label for the LFIB entry is being created. |
| next_hop=a.b.c.d | IP address of the next hop for the destination. |
| interface | The outgoing interface through which a packet will be sent. |
| get ip adj | Shows that the IP adjacency to use in the LFIB entry is being determined. |
| get tag adj | Shows that the label switching adjacency to use for the LFIB entry is being determined. |
| addr = a.b.c.d | The IP address of the adjacency. |
| is_p2p=x | If x is 1, this is a point-to-point adjacency. If x is 0, it is not. |
| fibidb = s | Indicates the interface of the adjacency. |
| linktype = x | The link type of the adjacency: 7 = LINK_IP 79 = LINK_TAG |
| sizes x,y,z | Indicates the following values: x = length of macstring y = length of tag encapsulation z = tag MTU |
| type = x | Tag encapsulation type: 0 = normal 1 = TCATM 2 = TSP tunnel |
| idb:s | Indicates the outgoing interface. |
| update_mac | Shows that the macstring of the adjacency is being updated. |

Table 7 describes the special labels, which sometimes appear in the debug output, and their meanings.

Table 7 Special Labels Appearing in Debug Command Output

| Special Label | Meaning |
|----------------------|--|
| Unassn-Initial value | No label assigned yet. |
| Unused | This destination does not have a label (for example, a BGP route). |
| Withdrn | The label for this destination has been withdrawn. |
| Unkn | This destination should have a label, but it is not yet known. |
| Get_res | A recursive route that will get a label when resolved. |
| Exp_null | Explicit null label—used over TC-ATM. |
| Imp_null | Implicit null label—for directly connected routes. |
| Tun_hd | Identifies head of TSP tunnel. |

| Related Commands | Command | Description |
|-------------------------|-------------------------------|---|
| | debug mpls lfib cef | Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed. |
| | debug mpls lfib lsp | Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed. |
| | debug mpls lfib state | Traces what happens when label switching is enabled or disabled. |
| | debug mpls lfib struct | Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label-rewrites, and label-info data. |

debug mpls lfib lsp

To print detailed information about label rewrites being created and deleted as TSP tunnels are added or removed, use the **debug mpls lfib lsp** EXEC command. The **no** form of this command disables debugging output.

debug mpls lfib lsp

no debug mpls lfib lsp

Syntax Description This command has no keywords or arguments.

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Examples

The following is sample output generated from the **debug mpls lfib lsp** command:

```
Router# debug mpls lfib lsp
```

```
TSP-tunnel related TFIB services debugging is on
```

```
TFIB: tagtun,next hop=10.93.72.13,inc=35,outg=1,idb=Et4/0/3
TFIB: tsptunnel:next hop=10.93.72.13,inc=35,outg=Imp_null,if_number=7
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun tag chg linec,fiblc=0,in tg=35,o tg=1,if=7,nh=10.93.72.13
TFIB: tagtun,next hop=10.92.0.7,inc=36,outg=1,idb=Et4/0/2
TFIB: tsptunnel:next hop=10.92.0.7,inc=36,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=36,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun tag chg linec,fiblc=0,in tg=36,o tg=1,if=6,nh=10.92.0.7
TFIB: tagtun_delete, inc = 36
tagtun tag del linec,itag=12
TFIB: tagtun_delete, inc = 35
tagtun tag del linec,itag=12
TFIB: tagtun,next hop=10.92.0.7,inc=35,outg=1,idb=Et4/0/2
TFIB: tsptunnel:next hop=10.92.0.7,inc=35,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun tag chg linec,fiblc=0,in tg=35,o tg=1,if=6,nh=10.92.0.7
```

On VIP:

```
TFIB: tagtun chg msg,in tg=35,o tg=1,nh=10.93.72.13,if=7
TFIB: tsptunnel:next hop=10.93.72.13,inc=35,outg=Imp_null,if_number=7
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=36,o tg=1,nh=10.92.0.7,if=6
TFIB: tsptunnel:next hop=10.92.0.7,inc=36,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=36,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=35,o tg=1,nh=10.93.72.13,if=7
TFIB: tsptunnel:next hop=10.93.72.13,inc=35,outg=Imp_null,if_number=7
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=36,o tg=1,nh=10.92.0.7,if=6
TFIB: tsptunnel:next hop=10.92.0.7,inc=36,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=36,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=35,o tg=1,nh=10.92.0.7,if=6
TFIB: tsptunnel:next hop=10.92.0.7,inc=35,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
```

Table 8 describes the significant fields in the sample display shown above.

Table 8 debug mpls lfib lsp Command Field Descriptions

| Field | Description |
|---------------------------|--|
| tagtun | Name of routine entered. |
| next hop=x.y.z.w | Next hop for the tunnel being created. |
| inc=x | Incoming label for this hop of the tunnel being created. |
| outg=x | Outgoing label (1 means Implicit Null label). |
| idb=s | Outgoing interface for the tunnel being created. |
| if_number=7 | Interface number of the outgoing interface. |
| tsptunnel | Name of the routine entered. |
| tsptun update loadinfo | The procedure being performed. |
| tag=x | Incoming label of LFIB slot whose loadinfo is being updated. |
| loadinfo_reqd=x | Shows whether a loadinfo is expected for this entry (nonrecursive accounting is on). |

Table 8 debug mpls lfib lsp Command Field Descriptions (continued)

| Field | Description |
|----------------------|---|
| no new loadinfo | No change required in loadinfo. |
| no old loadinfo | No previous loadinfo available. |
| tagtun tag chg linec | Line card is being informed of the TSP tunnel. |
| fiblc=x | Indicates which line card is being informed (0 means all). |
| in tg=x | Indicates incoming label of new TSP tunnel. |
| o tg=x | Indicates outgoing label of new TSP tunnel. |
| if=x | Indicates outgoing interface number. |
| nh=x.y.w.z | Indicates next hop IP address. |
| tagtun_delete | Indicates that a procedure is being performed: delete a TSP tunnel. |
| tagtun tag del linec | Informs the line card of TSP tunnel deletion. |
| tagtun chg msg | Indicates that line card has received a message to create a TSP tunnel. |

Related Commands

| Command | Description |
|-------------------------------|---|
| debug mpls lfib cef | Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed. |
| debug mpls lfib state | Traces what happens when label switching is enabled or disabled. |
| debug mpls lfib struct | Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label-rewrites, and label-info data. |

debug mpls lfib state

To trace what happens when label switching is enabled or disabled, use the **debug mpls lfib state EXEC** command. The **no** form of this command disables debugging output.

debug mpls lfib state

no debug mpls lfib state

Syntax Description This command has no keywords or arguments.

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Usage Guidelines Use this command when you wish to trace what happens to the LFIB when you issue the **mpls ip** or the **mpls tsp-tunnel** command.

Examples The following is sample output generated from the **debug mpls lfib state** command:

```
Router# debug mpls lfib state

TFIB enable/disable state debugging is on
TFIB: Upd tag sb 6(status:0xC1,tmtu:1500,VPI:1-1 VC=0/32,et:0/0/0),lc 0x0
TFIB: intf status chg: idb=Et4/0/2,status=0xC1,oldstatus=0xC3
TFIB: interface dyntag change,change in state to Ethernet4/0/2
TFIB: enable entered, table exists,enabler type=0x2
TFIB: enable, TFIB already enabled, types now 0x3,returning
TFIB: enable entered, table exists,enabler type=0x1
TFIB: disable entered, table exists,type=0x1

TFIB: cleanup: tfib[32] still non-0

On linecard only:

TFIB: disable lc msg recvd, type=0x1
TFIB: Ethernet4/0/1 fibidb subblock message received
TFIB: enable lc msg recvd, type=0x1
TFIB: Tunnel301 set encapsfix to 0x6016A97C
```

Table 9 describes the significant fields in the sample display shown above.

Table 9 debug mpls lfib state Command Field Descriptions

| Field | Description |
|---|---|
| LFIB | Identifies the source of the message as the LFIB subsystem. |
| Upd tag sb x | Shows that the status of the “xth” label switching subblock is being updated, where x is the interface number. There is a label switching subblock for each interface on which label switching has been enabled. |
| (status:0xC1,tmtu:1500 ,VPI:1-1VC=0/32, et:0/0/0),lc 0x0) | Identifies the values of the fields in the label switching subblock: status byte maximum transmission unit (<i>tmtu</i>) range of ATM VPs control VP control VC (if this is a TC-ATM interface) encapsulation type (<i>et</i>) encapsulation information tunnel interface number (<i>lc</i>) line card number to which the update message is being sent (0 means all line cards) |
| intf status chg | Indicates that there was an interface status change. |
| idb=Et4/0/2 | Identifies the interface whose status changed. |
| status=0xC1 | Shows the new status bits in the label switching subblock of the idb. |
| oldstatus=0xC3 | Shows the old status bits before the change. |
| interface dyntag change, change in state to Ethernet4/0/2 | Indicates that there was a change in the dynamic label status for the particular interface. |
| enable entered | Shows that the code that enables the LFIB was invoked. |
| TFIB already enabled | Shows that the LFIB was already enabled when this call was made. |
| table exists | Shows that an LFIB table had already been allocated in a previous call. |
| cleanup: tfib[x] still non-0 | Indicates that the LFIB is being deleted, but that slot x is still active. |
| disable lc mesg recvd, type=0x1 | Shows that a message to disable label switching type 1 (dynamic) was received by the line card. |
| disable entered, table exists, type=0x1 | Shows that a call to disable dynamic label switching was issued. |
| Ethernet4/0/1 fibidb subblock message received | Shows that a message giving fibidb status change was received on the line card. |
| enable lc msg recvd, type=0x1 | Shows that the line card received a message to enable label switching type 1 (dynamic). |
| Tunnel301 set encapsfix to 0x6016A97C | Shows that fibidb Tunnel301 on the line card received an encapsulation fixup. |
| types now 0x3, returning | Shows the value of the bitmask indicating the type of label switching enabled on the interface: 0x1—means dynamic label switching. 0x2—means tsp-tunnels. 0x3—means both. |

| Related Commands | Command | Description |
|------------------|-------------------------------|---|
| | debug mpls lfib cef | Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed. |
| | debug mpls lfib state | Traces what happens when label switching is enabled or disabled. |
| | debug mpls lfib lsp | Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed. |
| | debug mpls lfib struct | Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label-rewrites, and label-info data. |

debug mpls lfib struct

To trace the allocation and freeing of LFIB-related data structures, such as the LFIB itself, label-rewrites, and label-info data, use the **debug mpls lfib struct** EXEC command. The **no** form of this command disables debugging output.

debug mpls lfib struct

no debug mpls lfib struct

Syntax Description This command has no keywords or arguments.

Defaults This command has no default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|----------|--|
| | 11.1CT | This command was introduced. |
| | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |

Examples

The following is sample output generated from the **debug mpls lfib struct** command:

```
Router# debug mpls lfib struct

TFIB data structure changes debugging is on

TFIB: delete tag rew, incoming tag 32
TFIB: remove from tfib,inc tag=32
TFIB: set loadinfo,tag=32,no old loadinfo,no new loadinfo
TFIB: TFIB not in use.  Checking for entries.
TFIB: cleanup: tfib[0] still non-0
TFIB: remove from tfib,inc tag=Tun_hd
TFIB: set loadinfo,tag=Exp_null,no old loadinfo,no new loadinfo
TFIB: TFIB freed.
TFIB: enable, TFIB allocated, size 4024 bytes, maxtag = 500
TFIB: create tag rewrite: inc Tun_hd,outg Unkn
TFIB: add to tfib at Tun_hd, first in circular list, mac=0,enc=0
TFIB: delete tag rew, incoming tag Tun_hd
TFIB: remove from tfib,inc tag=Tun_hd
TFIB: set loadinfo,tag=Exp_null,no old loadinfo,no new loadinfo
TFIB: create tag rewrite: inc Tun_hd,outg Unkn
TFIB: add to tfib at Tun_hd, first in circular list, mac=0,enc=0
TFIB: create tag rewrite: inc 26,outg Unkn
TFIB: add to tfib at 26, first in circular list, mac=0,enc=0
TFIB: add to tfib at 27, added to circular list, mac=0,enc=0
TFIB: delete tag rew, incoming tag Tun_hd
TFIB: remove from tfib,inc tag=Tun_hd
TFIB: set loadinfo,tag=Exp_null,no old loadinfo,no new loadinfo
TFIB: add to tfib at 29, added to circular list, mac=4,enc=8
TFIB: delete tag rew, incoming tag 29
TFIB: remove from tfib,inc tag=29
```

Table 10 describes the significant fields in the sample display shown above.

Table 10 debug mpls lfib struct Command Field Descriptions

| Field | Description |
|--|---|
| TFIB | The subsystem issuing the message. |
| delete tag rew | A label_rewrite is being freed. |
| remove from tfib | A label rewrite is being removed from the LFIB. |
| inc tag=s | The incoming label of the entry being processed. |
| set loadinfo | The loadinfo field in the LFIB entry is being set (used for nonrecursive accounting). |
| tag=s | The incoming label of the entry being processed. |
| no old loadinfo | The LFIB entry did not have a loadinfo before. |
| no new loadinfo | The LFIB entry should not have a loadinfo now. |
| TFIB not in use. Checking for entries. | Label switching has been disabled and the LFIB is being freed up. |
| cleanup: tfib[x] still non-0 | The LFIB is being checked for any entries in use, and entry x is the lowest numbered slot still in use. |
| TFIB freed | The LFIB table has been freed. |
| enable, TFIB allocated, size x bytes, maxtag = y | Label switching has been enabled and an LFIB of x bytes has been allocated. The largest legal label is y. |

Table 10 debug mpls lfib struct Command Field Descriptions (continued)

| Field | Description |
|------------------------|---|
| create tag rewrite | A label_rewrite is being created. |
| inc s | The incoming label. |
| outg s | The outgoing label. |
| add to tfib at s | A label_rewrite has been placed in the LFIB at slot s. |
| first in circular list | This LFIB slot had been empty and this is the first rewrite in the list. |
| mac=0,enc=0 | Length of the mac string and total encapsulation length, including labels. |
| added to circular list | A label_rewrite is being added to an LFIB slot which already had an entry. This rewrite is being inserted in the circular list. |

Related Commands

| Command | Description |
|------------------------------|---|
| debug mpls lfib cef | Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed. |
| debug mpls lfib lsp | Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed. |
| debug mpls lfib state | Traces what happens when label switching is enabled or disabled. |

debug mpls packets

To display labeled packets switched by the host router, use the **debug mpls packets** EXEC command. The **no** form of this command disables debugging output.

debug mpls packets [*interface*]

no debug mpls packets [*interface*]

| Syntax Description | <i>interface</i> (Optional.) Interface or subinterface name. | | | | | | |
|---------------------------|---|---------|--------------|--------|------------------------------|----------|--|
| Defaults | Displays all labeled packets regardless of interface. | | | | | | |
| Command Modes | Privileged EXEC | | | | | | |
| Command History | <table border="1"> <thead> <tr> <th>Release</th> <th>Modification</th> </tr> </thead> <tbody> <tr> <td>11.1CT</td> <td>This command was introduced.</td> </tr> <tr> <td>12.1(3)T</td> <td>This command was modified to reflect MPLS IETF syntax and terminology.</td> </tr> </tbody> </table> | Release | Modification | 11.1CT | This command was introduced. | 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. |
| Release | Modification | | | | | | |
| 11.1CT | This command was introduced. | | | | | | |
| 12.1(3)T | This command was modified to reflect MPLS IETF syntax and terminology. | | | | | | |

Usage Guidelines The optional *interface* parameter restricts the display to only those packets received or transmitted on the indicated interface.



Note

Use this command with care because it generates output for every packet processed. Furthermore, enabling this command causes fast and distributed label switching to be disabled for the selected interfaces. To avoid adversely affecting other system activity, use this command only when traffic on the network is at a minimum.

Examples The following is sample output from the **debug mpls packets** command:

```
Router# debug mpls packets

TAG: Hs3/0: rcvd: CoS=0, TTL=254, Tag(s)=27
TAG: Hs0/0: xmit: (no tag)

TAG: Hs0/0: rcvd: CoS=0, TTL=254, Tag(s)=30
TAG: Hs3/0: xmit: CoS=0, TTL=253, Tag(s)=27
```

Table 11 describes the significant fields in the sample display shown above.

Table 11 debug mpls packets Command Field Descriptions

| Field | Description |
|----------|---|
| Hi0/0 | The identifier for the interface on which the packet was received or transmitted. |
| rcvd | Packet received. |
| xmit | Packet transmitted. |
| CoS | Class of Service field from the packet label header. |
| TTL | Time To Live field from the packet label header. |
| (no tag) | Last label popped off the packet and transmitted unlabeled. |
| Tag(s) | A list of labels on the packet, ordered from the top of the stack to the bottom. |

Related Commands

| Command | Description |
|-----------------------------------|---|
| show mpls forwarding-table | Displays the contents of the MPLS forwarding table. |

Glossary

ATM edge LSR—A router that is connected to the ATM-LSR cloud through LC-ATM interfaces. The ATM edge LSR adds labels to unlabeled packets and strips labels from labeled packets.

ATM-LSR—A label switch router with a number of LC-ATM interfaces. The router forwards the cells among these interfaces using labels carried in the VPI/VCI field of the ATM cell header.

CoS—Class of service. A feature that provides scalable, differentiated types of service across an MPLS network.

IP precedence—A 3-bit value in a ToS byte used for assigning precedence to IP packets.

label—A short fixed-length label that tells switching nodes how to forward data (packets or cells).

label-controlled ATM interface (LC-ATM interface)—An interface on a router or switch that uses label distribution procedures to negotiate label VCs.

label edge router (LER)—A router that performs label imposition.

label imposition—The action of putting the first label on a packet.

label switch—A node that forwards units of data (packets or cells) on the basis of labels.

label-switched path (LSP)—A sequence of hops (Router 0...Router n) in which a packet travels from R0 to Rn by means of label switching mechanisms. A label-switched path can be chosen dynamically, based on normal routing mechanisms, or it can be configured manually.

label-switched path (LSP) tunnel—A configured connection between two routers, in which label switching techniques are used for packet forwarding.

label switching router (LSR)—A Layer 3 router that forwards a packet based on the value of a label encapsulated in the packet.

label VC (LVC)—An ATM virtual circuit that is set up through ATM LSR label distribution procedures.

LFIB—Label Forwarding Information Base. The data structure used by switching functions to switch labeled packets.

LIB—Label information base. A database used by an LSR to store labels learned from other LSRs, as well as labels assigned by the local LSR.

MPLS—Multiprotocol label switching. An emerging industry standard that defines support for MPLS forwarding of packets along normally routed paths (sometimes called MPLS hop-by-hop forwarding).

QoS—Quality of service. A measure of performance for a transmission system that reflects its transmission quality and service availability.

tailend—The downstream, received end of a tunnel.

TDP—Tag Distribution Protocol. The protocol used to distribute label bindings to LSRs.

traffic engineering—The techniques and processes used to cause routed traffic to travel through the network on a path other than the one that could have been chosen if standard routing methods had been applied.

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.

VPN—Virtual private network. Enables IP traffic to use tunneling to travel securely over a public TCP/IP network.

