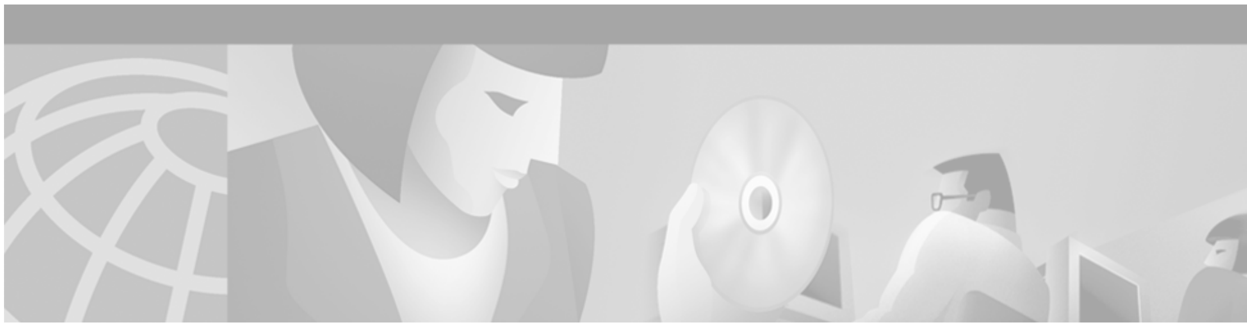


MPLS Traffic Engineering (TE) MIB

This document describes the Simple Network Management Protocol (SNMP) agent support in Cisco IOS for Multiprotocol Label Switching (MPLS) traffic engineering management, as implemented in the MPLS Traffic Engineering MIB (MPLS TE MIB).

Multiprotocol Label Switching (MPLS) traffic engineering capabilities in Cisco IOS enable an MPLS backbone to replicate and expand upon the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks.

Traffic engineering 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 traffic engineering facilities built into Cisco IOS provide a feature-rich, integrated approach to managing the large volumes of traffic that typically flow through wide area networks (WANs). The MPLS traffic engineering 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.

Feature Overview

SNMP agent code operating in conjunction with the MPLS TE MIB enables a standardized, SNMP-based approach to be used in managing the MPLS traffic engineering features in Cisco IOS.

The MPLS TE MIB is based on the IETF draft MIB entitled *draft-ietf-mpls-te-mib-05.txt*, which includes objects describing features that support MPLS traffic engineering. This IETF draft MIB, which undergoes revisions from time to time, is being evolved toward becoming a standard. Accordingly, Cisco's implementation of the MPLS TE MIB is expected to track the evolution of the IETF draft MIB.

Slight differences between the IETF draft MIB and the implementation of the traffic engineering capabilities within Cisco IOS require some minor translations between the MPLS TE MIB and the internal data structures of Cisco IOS. These translations are accomplished by means of 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 IOS.

The SNMP objects defined in the MPLS TE MIB can be viewed 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.

Benefits of MPLS TE MIB

The MPLS Traffic Engineering MIB provides the following benefits:

- Provides a standards-based SNMP interface for retrieving information about MPLS traffic engineering.
- Provides information about the traffic flows on MPLS traffic engineering tunnels.
- Presents MPLS traffic engineering tunnel routes, including the configured route, the IGP calculated route, and the actual route traversed.
- Provides information, in conjunction with the Interfaces MIB (ifMIB), about how a tunnel was rerouted in the event of a link failure.
- Provides information about the configured resources used for an MPLS traffic engineering tunnel.

Structure of MPLS TE MIB

The SNMP agent code supporting the MPLS TE MIB follows the existing model for such code in Cisco IOS and is, in part, generated by the Cisco IOS 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 IOS, consists of four layers:

- Platform independent layer—This layer is generated primarily by the IOS MIB development tool set and incorporates platform and implementation independent functions. The IOS 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 IOS MIB development 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 IOS, such as searching through data structures.
- API and data structures layer—This layer contains the data structures or APIs within Cisco IOS that are retrieved or called in order to set or retrieve SNMP management information.

Restrictions

The following restrictions apply to the MPLS TE MIB for Cisco IOS Release 12.0(16)S:

- Supports read-only (RO) permission for MIB objects.
- Contains no configuration support by means of SET functions; only SNMP GET, GETNEXT, and GETBULK retrieval functions are supported in this release.
- Contains no support for TE MIB notifications; Interfaces MIB (ifMIB) notifications are sent when tunnel interfaces go up or down.
- Contains no support for Guaranteed Bandwidth Traffic Engineering (GBTE) or Auto Bandwidth features.

Related Features and Technologies

The MPLS TE MIB feature is used in conjunction with the following:

- Standards-based SNMP network management application
- Multiprotocol label switching (MPLS)
- MPLS traffic engineering
- MPLS label switching router MIB (MPLS-LSR-MIB)

Related Documents

For descriptions of other MPLS-based functionality, consult the following documentation:

- *MPLS Label Distribution Protocol Enhancements*
- *MPLS Label Switching Router MIB*
- *MPLS Scalability Enhancements for LSC and ATM LSR*
- *Automatic Bandwidth Adjustment for MPLS Traffic Engineering Tunnels*
- *Scalability Enhancements for MPLS Traffic Engineering*
- *MPLS Class of Service Enhancements*
- *MPLS Traffic Engineering Access List Node Exclusion*
- *RFC 2233 Interfaces MIB*

Supported Platforms

The MPLS TE MIB is supported on the following platforms:

- Cisco 12000 series Internet routers
- Cisco 7500 series routers

Supported Standards, MIBs, and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

The MPLS TE MIB is supported on Cisco IOS Release 12.0(16)S.

To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB web site on Cisco Connection Online (CCO) at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

RFCs

The structure and content of the MPLS TE MIB is in full conformance with the provisions of Section 10 of RFC 2026.

The Internet Engineering Task force (IETF) document entitled *draft-ietf-mpls-te-mib-05.txt* defines and describes managed objects for traffic engineering based on multiprotocol label switching.

Supported Objects in MPLS TE MIB

The MPLS TE MIB contains numerous tables and object definitions that provide read-only SNMP management support for the MPLS traffic engineering features in Cisco IOS. The MPLS TE MIB conforms to Abstract Syntax Notation One (ASN.1), thus reflecting an idealized MPLS traffic engineering database.

Using any standard SNMP network management application, you can retrieve and display information from the MPLS TE MIB using GET operations; similarly, you can traverse information in the MIB database for display using GETNEXT operations.

The MPLS TE MIB tables and objects supported in this Cisco IOS release are listed below. Important MIB tables (those highlighted in bold type) are described briefly in accompanying text.

- mplsTunnelConfigured—Presents the total number of tunnel configurations that are defined on this node.
- mplsTunnelActive—Presents the total number of LSPs that are defined on this node.
- mplsTunnelTEDistProto—The IGP distribution protocol in use.
- mplsTunnelMaxHops—The maximum number of hops any given tunnel may 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 label switched paths (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.

  The entries in the mplsTunnelTable are listed and described below. Each description carries an alphabetic suffix (a), (b), or (c), if/as appropriate, to indicate the applicability of the entry:

  a. For tunnel head configurations only
  b. For LSPs only
  c. For both tunnel head configurations and LSPs

  Pointers in the tunnel table refer to corresponding entries in other MIB tables. By means of these pointers, you can find an entry in the *mplsTunnelTable* and follow a pointer to other tables for additional information. The pointers include the following: *mplsTunnelResourcePointer*, *mplsTunnelHopTableIndex*, *mplsTunnelARHopTableIndex*, and *mplsTunnelCHopTableIndex*.

  The tunnel table is indexed by tunnel id, tunnel instance, tunnel source address, and tunnel destination address.

  – 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—CLI name for the tunnel interfaces (a).
  – mplsTunnelDescr—Descriptive name for tunnel configurations and LSPs (c).
  – mplsTunnelIsIf—Indicates whether or not the entry represents an interface (c).
  – mplsTunnelIfIndex—Provides the index of the tunnel interface within the ifMIB (a).

- mplsTunnelXCPointer—(For midpoints only – no tails); provides a pointer for the LSP within the mplsXCTable of the MPLS LSR MIB (b).
- mplsTunnelSignallingProto—Identifies the signalling protocol used by tunnels (c).
- mplsTunnelSetupPrio—Indicates the setup priority of the tunnel (c).
- mplsTunnelHoldingPrio—Indicates the holding priority of the tunnel (c).
- mplsTunnelSessionAttributes—Indicates the session attributes (c).
- mplsTunnelOwner—Indicates the tunnel owner (c).
- mplsTunnelLocalProtectInUse—Not implemented (c).
- mplsTunnelResourcePointer—Provides a pointer into the Resource Table (b).
- mplsTunnelInstancePriority—Not implemented (b).
- mplsTunnelHopTableIndex—Provides an index into the Hop Table (a).
- mplsTunnelARHopTableIndex—Provides an index into the AR Hop Table (b).
- mplsTunnelCHopTableIndex—Provides an index into the C Hop Table (b).
- mplsTunnelPrimaryTimeUp—The amount of time that the current path has been up (a).
- mplsTunnelPathChanges—Number of times a tunnel has been resignalled (a).
- mplsTunnelLastPathChange—The amount of time since the last path resignalling occurred (a).
- mplsTunnelCreationTime—Time stamp when the tunnel was created (a).
- mplsTunnelStateTransitions—The 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 currently being used for the tunnel's path. If no path option is active, this object will be 0 (a).
- mplsTunnelRole—(c).
- mplsTunneltotalUptime—Amount of time that the tunnel has been operationally up (a).
- mplsTunnelInstanceUptime—Not implemented (b).
- mplsTunnelAdminStatus—Indicates the administrative status of a tunnel (c).
- mplsTunnelOperStatus—Indicates the 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—(c).

• mplsTunnelHopListIndexNext

• **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.

  - mplsTunnelHopListIndex

  - mplsTunnelHopIndex
  - mplsTunnelHopAddrType
  - mplsTunnelHopIpv4Addr
  - mplsTunnelHopIpv4PrefixLen
  - mplsTunnelHopIpv6Addr
  - mplsTunnelHopIpv6PrefixLen
  - mplsTunnelHopAsNumber
  - mplsTunnelHopLspId
  - mplsTunnelHopType
  - mplsTunnelHopRowStatus
  - mplsTunnelHopStorageType
- mplsTunnelResourceIndexNext
- **mplsTunnelResourceTable**—Entries in this table correspond to the "Tspec" information displayed when you execute the **show mpls traffic-eng tunnels** command. These entries exist only for label switched paths (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.
  - mplsTunnelResourceIndex
  - mplsTunnelResourceMaxRate
  - mplsTunnelResourceMeanRate
  - mplsTunnelResourceMaxBurstSize
  - mplsTunnelResourceRowStatus
  - mplsTunnelResourceStorageType
- **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 RSVP. You can view the information in this table by executing the **show mpls traffic-eng tunnels** command. Viewing such record route information, however, requires that you first execute the **tunnel mpls traffic-eng record-route** command on the host to enable the record route function.

  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.
  - mplsTunnelARHopListIndex
  - mplsTunnelARHopIndex
  - mplsTunnelARHopAddrType
  - mplsTunnelARHopIpv4Addr
  - mplsTunnelARHopIpv4PrefixLen
  - mplsTunnelARHopIpv6Addr
  - mplsTunnelARHopIpv6PrefixLen
  - mplsTunnelARHopAsNumber

  - mplsTunnelARHopType

- **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 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 in order to complete the path. If the user specifies a complete, explicit path, the computed hop table matches the user-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.

  - mplsTunnelCHopListIndex
  - mplsTunnelCHopIndex
  - mplsTunnelCHopAddrType
  - mplsTunnelCHopIpv4Addr
  - mplsTunnelCHopIpv4PrefixLen
  - mplsTunnelCHopIpv6Addr
  - mplsTunnelCHopIpv6PrefixLen
  - mplsTunnelCHopAsNumber
  - mplsTunnelCHopType

mplsTunnelPerfTable—The tunnel performance table, which augments the **mplsTunnelTable** described above, provides packet and byte counters for each tunnel. This table contains the following packet/byte counters:

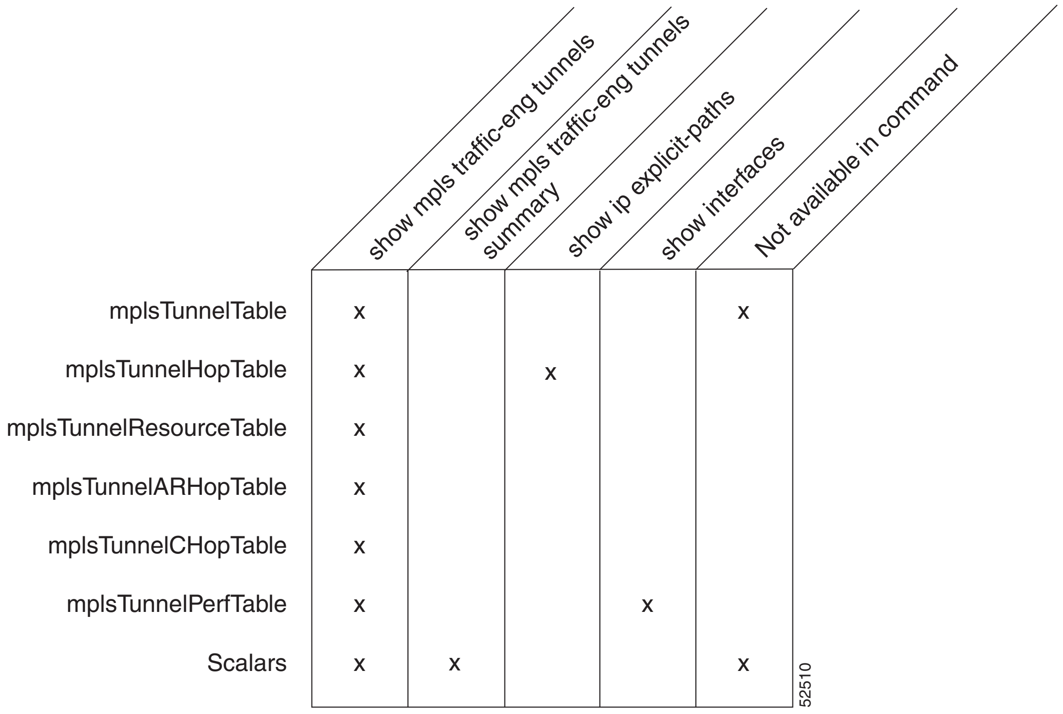
  - mplsTunnelPerfPackets—This packet counter works only for tunnel heads.
  - mplsTunnelPerfHCPackets—This packet counter works only for tunnel heads.
  - mplsTunnelPerfErrors—This packet counter works only for tunnel heads.
  - mplsTunnelPerfBytes—This byte counter works for tunnel heads and tunnel midpoints, but not for tunnel tails.
  - mplsTunnelPerfHCBytes—This byte counter works for tunnel heads and tunnel midpoints, but not for tunnel tails.

- mplsTunnelTrapEnable

CLI Access to MPLS TE MIB Information

Figure 1 shows specific CLI commands that can be used to retrieve information from specific tables in the MPLS TE MIB. As noted in this table, some information in the MPLS TE MIB is not retrievable by means of CLI commands.

Figure 1 CLI 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 TE MIB

This section describes an efficient procedure to aid the user in efficiently retrieving information about traffic engineering tunnels. Such information can be quite useful in large networks which often contain many traffic engineering tunnels.

First, begin by traversing across a single column of the *mplsTunnelTable*, such as *mplsTunnelName*. This action provides the indexes of every tunnel configuration, as well as any label switched paths (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*.

Furthermore, the *mplsTunnelTable* provides pointers to other tables for each tunnel. The column *mplsTunnelResourcePointer*, for example, provides an object ID (OID) that can be used to access resource allocation information in the *mplsTunnelResourceTabl*e. 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, information pertaining to the hops of that tunnel configuration can be retrieved.

Also, since tunnels are treated as interfaces, the tunnel table column (*mplsTunnelIfInde*x), provides an index into the Interfaces MIB that can be used to retrieve interface specific information about a tunnel.

Configuration Tasks

This section describes the following MPLS TE MIB configuration tasks:

- Enabling the SNMP Agent (required)
- Verifying the Status of the SNMP Agent (optional)

Enabling the SNMP Agent

The SNMP agent for the MPLS TE MIB is disabled by default.

To enable the SNMP agent for the MPLS TE MIB, perform the steps shown in the following table:

	Command	Purpose
Step 1	`Prompt#` **`telnet`** ***`xxx.xxx.xxx.xxx`***	Telnets to the router identified by the specified IP address (represented as *xxx.xxx.xxx.xxx*).
Step 2	`Router#` **`enable`**	Enters the enable mode.
Step 3	`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 Step 4. If any SNMP information is displayed, you can modify the information or change it as needed.
Step 4	`Router#` **`config terminal`**	Enters the global configuration mode.
Step 5	`Router(config)#` **`snmp-server community xxxxxx`** ***`RO`***	Enables the read-only (*RO*) community string, where *xxxxxx* represents the read-only community string
Step 6	`Router(config)#` **`exit`**	Exits the global configuration mode and returns you to the privileged EXEC mode.
Step 7	`Router#` **`write memory`**	Writes the modified configuration to nonvolatile memory (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 steps shown in the following table:

Step 1 Telnet to the target device:

```
Router# telnet xxx.xxx.xxx.xxx
```

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

Step 2 Enable SNMP on the target device:

```
Router# enable
```

Step 3 Display the running configuration on the target device and examine the output for displayed 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 which takes the form shown above verifies that SNMP has been enabled on that device.

Configuration Examples

The following example shows how to enable an SNMP agent on a host network device:

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

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 *publi*c.

```
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
```

Command Reference

This section documents the following CLI command for use with this Cisco IOS release:

- **snmp-server community**

Other CLI commands used with the MPLS TE MIB feature are documented in the Cisco IOS Release 12.0 command reference publications.

snmp-server community

Use the **snmp-server community** global configuration command on your host network management station (NMS) to configure read-only Simple Network Management Protocol (SNMP) community strings for the MPLS TE MIB. Use the **no snmp-server community** command to change the community string to its default value.

snmp-server community *string* [**view** *view-name*] [**ro**] [*number*]

[**no**] **snmp-server community** *string*

Syntax Description

string	A community string consists of 1 to 32 alphanumeric characters and acts like a password, permitting access to SNMP functionality on LSRs in your network. Blank spaces are not allowed in the community string.
view *view-name*	(Optional). The name of a previously-defined view delineating the objects available to the SNMP community.
ro	(Optional). This default keyword configures read-only (RO) access to MPLS TE MIBs on LSRs, thus limiting a network management station to retrieving objects from an MPLS TE MIB.
number	(Optional). This parameter is an integer from 1 to 99 specifying an access list of IP addresses that are allowed to use the community string to gain access to the SNMP v.1 agent.

Defaults

The default value of the read/write parameter is read-only (**ro**). The default value of the read-only community string is *public*, and the default value of the read-write community string is *private*.

Command Modes

Global configuration

Command History

Release	Modification
10.0	This command was introduced in this release.
12.0(16)S	This command was integrated into this release.

Usage Guidelines

The **no snmp-server** command disables both versions of SNMP (SNMPv1 and SNMPv2).

The first **snmp-server** command entered enables both versions of SNMP.

Examples

The following example shows how to set the read-write community string to *newstring*:

```
Router(config)# snmp-server community newstring rw
```

The following example shows how to assign the string *comaccess* to SNMPv1, allowing read-only access and specifying that IP access list 4 can use the community string:

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

The following example shows how to assign the string *mgr* to SNMPv1, allowing read-write access to the objects in the restricted view:

```
Router(config)# snmp-server community mgr view restricted rw
```

The following example shows how to remove the community *comaccess*.

```
Router(config)# no snmp-server community comaccess
```

The following example shows how to disable both versions of SNMP:

```
Router(config)# no snmp-server
```

Glossary

affinity bits—an MPLS traffic engineering tunnel's requirements on the attributes of the links it will cross. The tunnel's affinity bits and affinity mask must match up with the attributes of the various links carrying the tunnel.

call admission precedence—an 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 harder to route will have a higher priority, and can preempt tunnels that are easier 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.

head-end—The LSR at which the tunnel originates. The tunnel's "head" or tunnel interface will reside at this LSR as well.

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. **label-switched path (LSP)**—A sequence of hops (R0...Rn) in which a packet travels from R0 to Rn through label switching mechanisms. A -switched path can be chosen dynamically, based on normal routing mechanisms, or through configuration.

Management Information Base—See MIB.

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 SNMP. The value of a MIB object can be changed or retrieved using SNMP commands, usually by means of 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. An emerging industry standard that defines support for MPLS forwarding of packets along normally routed paths (sometimes called MPLS hop-by-hop forwarding).

Multiprotocol Label Switching traffic engineering—MPLS traffic engineering. A constraint-based routing algorithm for routing LSP tunnels.

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.

OSPF—Open shortest path first (OSPF). A link state routing protocol used for routing IP.

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. 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.

tail-end—The downstream, receive end of a tunnel.

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 to indicate that a significant event within Cisco IOS 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.

VCI—Virtual channel identifier. A 16-bit field in the header of an ATM cell. The VCI, together with the VPI (see below), is used to identify the next network VCL (see below) as the cell passes through a series of ATM switches on its way to its final destination.

VCC—Virtual channel connection. A VCC is a logical circuit consisting of VCLs (see below) that carries data between two end points in an ATM network. Sometimes called a virtual circuit connection.

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 (see above), is used to identify the next network VCL (see above) as the cell passes through a series of ATM switches on its way to its final destination.