# MPLS-TP MIB

**Note** This chapter is not applicable on the ASR 900 RSP3 Module for the Cisco IOS XE Release 3.16.

The Multiprotocol Label Switching Transport Profile (MPLS-TP) allows you to meet your transport requirements as those requirements evolve from Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) time-division multiplexing (TDM) technologies to MPLS and Ethernet technologies. Currently, a strong momentum for MPLS-TP in terms of both rapid standards development and increasing market demand exists. MPLS-TP technologies have been recently requested by multiple service providers for packet transport primarily in the aggregation networks and access networks while the core network remains MPLS (MPLS-TP is being considered for core transport as well by one or two providers). Service providers aim at using MPLS-TP to support the following deployment scenarios: Ethernet services, mobile backhaul, Asynchronous Transfer Mode (ATM) aggregation replacement, video transport, and long haul transport.

MPLS TP MIB allows you to poll MPLS-TP configured nodes via Simple Network Management Protocol (SNMP) and monitor and manage the MPLS-TP network.

## Finding Feature Information

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

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

## Prerequisites for MPLS-TP MIB

- General knowledge of SNMP
- Software used to query Cisco devices via SNMP

## Restrictions for MPLS-TP MIB

- MPLS-TP MIB doesn’t specify any traps for TP and thus no trap support is provided.
- The MPLS-TP MIB module supports point-to-point, co-routed bi-directional tunnels.

## Information about MPLS-TP MIB

### Overview of MPLS-TP MIB

MPLS-TP MIB is part of the SNMP process. The MIB interacts with MPLS-TP functions to get the data required for objects and indices.

The following MIBs are implemented:

- CISCO-MPLS-TC-EXT-STD-MIB
- CISCO-MPLS-ID-STD-MIB
- CISCO-MPLS-LSR-EXT-STD-MIB
- CISCO-MPLS-TE-EXT-STD-MIB

### CISCO-MPLS-TC-EXT-STD-MIB

This MIB module contains textual conventions for MPLS-based transport networks.

| **Textual Convention** | **Description (from IETF draft)** |
|---|---|
| MplsGlobalId | This object contains the textual convention of the operator unique identifier (Global_ID). The Global_ID can contain the 2-octet or 4-octet value of the operator's Autonomous System Number (ASN). When the Global_ID is derived from a 2-octet AS number, the two high-order octets of this 4-octet identifier MUST be set to zero. ASN 0 is reserved. A Global_ID of zero means that no Global_ID is present. |

| | |
|---|---|
| MplsNodeId | The Node_ID is assigned within the scope of the Global_ID. The value 0 (or 0.0.0.0 in dotted decimal notation) is reserved and MUST NOT be used. When IPv4 addresses are in use, the value of this object can be derived from the LSR's /32 IPv4 loop back address. |
| MplsLocalId | This textual convention is used in accommodating the bigger size Global_Node_ID and/or ICC with lower size LSR identifier in order to index mplsTunnelTable. The Local Identifier is configured between 1 and 16777215, as the valid IP address range starts from 16777216 (01.00.00.00). This range is chosen to identify the mplsTunnelTable's Ingress/Egress.<br><br>LSR-id is the IP address or local identifier. If the configured range is not an IP address, the administrator is expected to retrieve the complete information (Global_Node_ID) from mplsNodeConfigTable. This way, the existing mplsTunnelTable is reused for bidirectional tunnel extensions for MPLS-based transport networks. |

## CISCO-MPLS-ID-EXT-STD-MIB

This MIB module contains generic object definitions for MPLS Traffic Engineering in transport networks.

| **Object** | **Description (from IETF draft)** |
|---|---|
| mplsGlobalId | This object allows the administrator to assign a unique operator identifier also called MPLS-TP Global_ID. |
| mplsNodeId | This object allows the operator or service provider to assign a unique MPLS-TP Node_ID. The Node_ID is assigned within the scope of the Global_ID. |

## MPLS LSR STD MIB

Existing Label Switch Router (LSR) MIB functions are used to fetch values for the tables below. For TP, an FPI type of FPI_IF4 is used for IPv4. Only IPv4 is supported in this release.

- **At the endpoints.** For each tunnel, there is one entry for mplsOutSegmentTable [RFC 3813] showing the outsegment label and one entry for mplsInSegmentTable [RFC 3813] for the working LSP. Similarly, an entry is shown to the protected LSP. The assumption is that both working and protected LSPs are configured. If only one working LSP and one protected LSP is configured, the entries are displayed accordingly. There are 2 entries per tunnel for mplsXCTable [RFC 3813] for a working LSP and similarly to a protected LSP.

- **At the midpoints.** For a co-routed bidirectional tunnel, a midpoint has forward and reverse LSPs configured. Thus, there are a pair of mplsInSegmentTable and mplsOutSegmentTable entries for the forward LSP and a pair of mplsInSegmentTable and mplsOutSegmentTable entries for the reverse LSP. If the working and protected LSPs are configured then the above listed entries are shown for both protected and working LSPs. For mplsXCTable, there are two entries—one for the forward LSP and one for the reverse LSP. If the config has working and protected LSPs configured, then the above listed mplsXCTable entries are shown for both protected and working LSPs.

- **Indexing for mplsOutSegmentTable, mplsInSegmentTable and mplsXCTable.** mplsXCTable is indexed by mplsXCIndex [RFC3813], mplsXCInSegmentIndex [RFC3813], and mplsXCOutSegmentIndex [RFC3813]. The mplsXCInSegmentIndex, which is the same as mplsInSegmentIndex, is a 4-byte octet string containing the local label. The mplsXCIndex for TP is represented in the octet string format. The FPI value of FPI_IF4 is taken from file lsd_common_issu_sensitive.enum. The FPI value of 3 is used for TP.
    - At the endpoint, mplsXCIndex is represented as an octet string that contains fpi_type, tunnel index, and the LSP identifier. The LSP identifier specifies if the LSP is working or protected. The LSP identifier can be of either two types: CFC_MPLS_CP_LSP_TYPE_WORKING - working LSP (integer value 2) or CFC_MPLS_CP_LSP_TYPE_PROTECT - protected LSP (integer value 3).

```
|----|          |----||----||----||----|          |----|

FPI = 3         Tunnel-id                       LSP_ident
```

**Note** Internally, Tunnel-id is used to get if_number (outgoing interface) and if_number is used to poll MFI where

```
|----|
```

equals 1 byte.

    - At the midpoint, mplsXCIndex is represented by an octet string that contains fpi_type and in-label. The Fpi_type value is 0 for label.

```
|----|           |----||----||----||----|

FPI = 0          Label
```

mplsXCOutSegmentIndex is the same as mplsOutSegmentIndex, which is the same as mplsXCIndex plus moi_index. The last two bytes in mplsOutSegmentIndex contain the MOI list index.

A new cfc_mpls_cp_lsrmib_rfc_get_tp_label_id MIB function will be created for the MIB team to fetch TP-related data.

| Object | Value and function used to get the value |
|---|---|
| **mplsOutSegmentTable** | |
| mplsOutSegmentIndex | This object contains the outsegment index as explained above. The cfc_mpls_cp_lsrmib_rfc_get_outseg_entry function is used to get this value. |
| mplsOutSegmentInterface | This object contains the outsegment interface that comes from the IDB. The cfc_mpls_cp_lsrmib_rfc_get_outseg_entry function is used to get this value. |

| | |
|---|---|
| mplsOutSegmentPushTopLabel | This object is set to D_mplsOutSegmentPushTopLabel_true. |
| mplsOutSegmentTopLabel | The lsrmib_get_top_label function is used to get this value. |
| mplsOutSegmentTopLabelPtr | Set to 0.0. |
| mplsOutSegmentNextHopAddrType | The value of mfi_out_info.nh.type provides the value of this object. |
| mplsOutSegmentNextHopAddr | The value of mfi_out_info.nh.ip_addr provides the value of this object. |
| mplsOutSegmentXCIndex | This object contains mplsXCIndex from mplsXCTable. The cfc_mpls_cp_lsrmib_rfc_get_xc_search_indices function is used to get this value. |
| mplsOutSegmentOwner | Will add a new a macro: LSRMIB_MPLS_FPI_IF4 and this will map to D_mplsOutSegmentOwner_tp. |
| mplsOutSegmentTrafficParamPtr | Always set to 0.0. |
| mplsOutSegmentRowStatus | D_mplsOutSegmentRowStatus_active |
| mplsOutSegmentStorageType | D_mplsInSegmentStorageType_volatile |
| **mplsOutSegmentPerfTable** | |
| mplsOutSegmentPerfOctets | mfi_out_info.bytes |
| mplsOutSegmentPerfPackets | mfi_out_info.packets |
| mplsOutSegmentPerfErrors | mfi_out_info.errors |
| mplsOutSegmentPerfDiscards | mfi_out_info.discards |
| mplsOutSegmentPerfHCOctets | Get from MFI. |
| mplsOutSegmentPerfDiscontinuityTime | lsrmib_get_discontinuity_time() |
| **mplsInSegmentTable** | |
| mplsInSegmentIndex | This object contains the insegmen index as explained above. The lsrmib_get_in_label_id function is used to get the value. |
| mplsInSegmentInterface | This is set to 0. |
| mplsInSegmentLabel | lsrmib_get_in_label_id function is used. |
| mplsInSegmentLabelPtr | Always set to 0.0. |
| mplsInSegmentNPop | Set to default value 1. |

| | |
|---|---|
| mplsInSegmentAddrFamily | Set to D_mplsInSegmentAddrFamily_ipV4. |
| mplsInSegmentXCIndex | This object contains mplsXCIndex. The cfc_mpls_cp_lsrmib_rfc_mfi_info_to_xc function is used to get this value. |
| mplsInSegmentOwner | D_mplsInSegmentOwner_other |
| mplsInSegmentTrafficParamPtr | 0.0 |
| mplsInSegmentRowStatus | D_mplsInSegmentRowStatus_active |
| mplsInSegmentStorageType | D_mplsInSegmentStorageType_volatile |
| **mplsInSegmentPerfTable** | |
| mplsInSegmentPerfOctets | mfi_out_info.bytes |
| mplsInSegmentPerfPackets | mfi_out_info.packets |
| mplsInSegmentPerfErrors | mfi_out_info.errors |
| mplsInSegmentPerfDiscards | mfi_out_info.discards |
| mplsInSegmentPerfHCOctets | Get from MFI. |
| mplsInSegmentPerfDiscontinuityTime | lsrmib_get_discontinuity_time() |
| **mplsXCTable** | |
| mplsXCIndex | cfc_mpls_cp_lsrmib_rfc_get_xc_search_indices function is used to get this value. |
| mplsXCInSegmentIndex | cfc_mpls_cp_lsrmib_rfc_get_xc_search_indices function is used to get this value. |
| mplsXCOutSegmentIndex | cfc_mpls_cp_lsrmib_rfc_get_xc_search_indices function is used to get this value. |
| mplsXCLSPId | cfc_mpls_cp_lsrmib_rfc_get_xc_search_indices is used to get this value. |
| mplsXCLabelStackIndex | This object contains the octet string 0.0. which indicates that no labels are to be stacked beneath the top label. |
| mplsXCOwner | RFC LSR MIB doesn't provide a specific value for TP. Thus, D_mplsXCOwner_other is used to fetch this value. |
| mplsXCRowStatus | Set to D_mplsXCRowStatus_active. |
| mplsXCStorageType | Set to D_mplsXCStorageType_volatile. |
| mplsXCAdminStatus | Set to D_mplsXCAdminStatus_up. |

| | |
|---|---|
| mplsXCOperStatus | Set to D_mplsXCOperStatus_up. |

## CISCO-MPLS-LSR-EXT-STD-MIB

**mplsXCExtEntry:** An entry in this table extends the cross connect information represented by an entry in the mplsXCTable through a sparse augmentation. The indices for this table are mplsXCIndex, mplsXCInSegmentIndex, and mplsXCOutSegmentIndex.

- **Midpoint.** At the midpoint there are 2 entries, one for the forward LSP and one for the reverse LSP. If both working and protected LSPs are configured, then there will be 2 entries for each of the LSPs.
- **Endpoint.** At the endpoint there are two entries in mplsXCExtTunnelPointer. If both working and protected LSPs are configured, then there will be 2 entries for each LSP.

| Object | Description | Value and function used to get the value |
|---|---|---|
| **mplsXCExtTunnelPointer** | This object indicates the back pointer to the tunnel entry segment. This object cannot be modified if mplsXCRowStatus for the corresponding entry in the mplsXCTable is active(1). | Both the entries (per tunnel) point to the same tunnel entry. A new function to fetch this information from TP will be created.<br>At the endpoint, the MIB code provides the tunnel number and the LSP identifier (working/protected) and expects in return from the TP the other two tunnel indices—the local ID for the source and the local ID for the destination of this tunnel.<br>At midpoint, the MIB code provides the incoming label and expects the TP to return the unique tunnel entry that provides the tunnel index, LSP instance, source-local-id, and destination-local-id. |

| | | |
|---|---|---|
| **mplsXCOppositeDirXCPtr** | This object indicates the pointer to the opposite direction XC entry. This object cannot be modified if mplsXCRowStatus for the corresponding entry in the mplsXCTable is active(1). | For the endpoint, there are two entries for this object. At the endpoint, the entry that represents the outgoing segment contains the mplsXCLspId entry that corresponds to the reverse direction in-label. The entry that corresponds to the in-label contains the mplsXCLspId representing the outgoing segment (so, in essence, contains the indices with FPI type 3 for the TP tunnel).<br><br>For the midpoint, there are two entries for this object. Each entry contains the mplsXCLspId representing the reverse direction in-label. |

## MPLS-TE-STD-MIB and MPLS Draft TE MIB

mplsTunnelTable from MPLS-TE-STD-MIB shows TP tunnel entries. For details on object description, refer to RFC 3812. Protected LSP is assumed to be configured for every working LSP.

TP configuration allows partial configuration. If an LSP is partially configured where destination node-id/global ID is not specified, then the local-id is set to 0.

- **Endpoint.** mplsTunnelTable has one entry per LSP.
- **Midpoint.** For the working LSP, mplsTunnelTable has one entry for the forward LSP and one entry for the reverse LSP. Similarly, if the protected LSP is configured, entries for the protected LSP are shown.

| **Object** | **Value and function used to get the value** |
|---|---|
| mplsTunnelIndex | At an endpoint, mplsTunnelIndex contains the source tunnel number.<br><br>At a midpoint, the mplsTunnelIndex contains the source tunnel number for the forward LSP and the destination tunnel number for the reverse LSP. |
| mplsTunnelInstance | This contains the LSP number. The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelIngressLSRId | At an endpoint, this contains the value of mplsNodeConfigLocalId for the source of the tunnel.<br><br>At a midpoint, it stores the mplsNodeConfigLocalId for the source of the tunnel for the forward LSP and mplsNodeConfigLocalId for the destination of the reverse LSP.<br><br>This value ranges between 1 and 16777215. The tp_get_tunnel_detail function is used to get this value. |

| | |
|---|---|
| mplsTunnelEgressLSRId | At an endpoint, this contains the value of mplsNodeConfigLocalId for the destination node of the tunnel.<br><br>At a midpoint, it stores the mplsNodeConfigLocalId for the destination of the tunnel for the forward LSP and mplsNodeConfigLocalId for the source of the tunnel for the reverse LSP.<br><br>This value ranges between 1 and 16777215. The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelName | This contains the tunnel name, applicable at both endpoint and midpoint. The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelDescr | This contains the tunnel description. The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelIsIf | This is always true because the TP tunnel is always an interface. |
| mplsTunnelIfIndex | This contains the tunnel ifindex. The tp_get_tunnel_detail function provides the IF number. The interface number can be used to get the interface index. |
| mplsTunnelOwner | This is set to D_mplsTunnelOwner_other. |
| mplsTunnelRole | The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelXCPointer | The cfc_mpls_cp_lsrmib_rfc_make_XC_pointer function is used. |
| mplsTunnelSignallingProto | None(1). The MPLS TP implementation on Cisco IOS does not have a control plane and there is no signaling protocol. |
| mplsTunnelSetupPrio | 0. By default, MPLS TP LSPs have 0 priority. |
| mplsTunnelHoldingPrio | 0. By default, MPLS TP LSPs have 0 priority. |
| mplsTunnelSessionAttributes | N/A. 0. |
| mplsTunnelLocalProtectInUse | This object indicates whether a protected LSP is being used. The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelResourcePointer | 0.0. Not supported. |
| mplsTunnelPrimaryInstance | This is used to indicate the LSP number of the working LSP. If the working LSP is not configured, then this shows a default value of 0. |

| | |
|---|---|
| mplsTunnelInstancePriority | N/A. 0. |
| mplsTunnelHopTableIndex | N/A. 0. |
| mplsTunnelPathInUse | N/A. 0. |
| mplsTunnelARHopTableIndex | N/A. 0. |
| mplsTunnelCHopTableIndex | N/A. 0. |
| mplsTunnelIncludeAnyAffinity | N/A. 0. |
| mplsTunnelIncludeAllAffinity | N/A. 0. |
| mplsTunnelTotalUpTime | The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelInstanceUpTime | The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelPrimaryUpTime | The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelPathChanges | N/A. 0. |
| mplsTunnelLastPathChange | N/A. |
| mplsTunnelCreationTime | The tp_get_tunnel_detail function is used to get this value. |
| mplsTunnelStateTransitions | N/A. 0. |
| mplsTunnelAdminStatus | At endpoint, the tp_get_tunnel_detail function is used to get this value.<br>At midpoint, this is set to "testing(3)" as the TP does not maintain admin status at the midpoint. |
| mplsTunnelOperStatus | At endpoint, the tp_get_tunnel_detail function is used to get this value.<br>At midpoint, this is set to "testing(3)" as the TP does not maintain oper status at the midpoint. |
| mplsTunnelRowStatus | D_mplsTunnelRowStatus_active |
| mplsTunnelStorageType | D_mplsTunnelStorageType_readOnly |
| **mplsTunnelPerfTable: This counter is not supported.** | |

## CISCO-MPLS-TE-EXT-STD-MIB

This MIB module contains generic object definitions for MPLS Traffic Engineering in transport networks.

| **Object** | **Description (as IETF draft defines it)** | **Value and function used to get the value** |
|---|---|---|
| **mplsNodeConfigTable** | | |
| mplsNodeConfigLocalId | This object allows the administrator to assign a unique local identifier to map Global_Node_ID. | This table is used to represent a node in a TP network. This object provides a unique local value for the node. The value of this object lies between 1 and 16777215.<br><br>The TP provides a new tp_get_node_detail function. This is used to get this object’s value. |
| mplsNodeConfigGlobalId | This object indicates the Global Operator Identifier. | This maps to the mpls_tp_global_id_t global_id field of the TP data structure.<br><br>tp_get_node_detail is used to get this object's value. |
| mplsNodeConfigNodeId | This object indicates the Node_ID within the operator. This object value should be zero when mplsNodeConfigIccId is configured with non-null value. | This object maps to mpls_tp_node_id_t node_id field of TP data structure.<br><br>The tp_get_node_detail function is used to get this object's value. |
| mplsNodeConfigIccId | This object allows the operator or service provider to configure a unique MPLS-TP ITU-T Carrier Code (ICC) either for Ingress ID or Egress ID. This object value should be zero when mplsNodeConfigGlobalId and mplsNodeConfigNodeId are assigned with a non-zero value. | This object is set to 0. Cisco IOS implementation only supports IP-compatible implementation. |
| mplsNodeConfigRowStatus | This object allows the administrator to create, modify, and/or delete a row in this table. | This is set to 'active'. |
| mplsNodeConfigStorageType | This variable indicates the storage type for this object. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row. | This is set to 'readonly' because write access to any object is not allowed. |
| **mplsNodeIpMapTable: This table is indexed by mplsNodeIpMapNodeId and mplsNodeIpMapLocalId** | | |
| mplsNodeIpMapGlobalId | This object indicates the Global_ID. | The tp_get_node_detail function is used to get this object's value. |

| | | |
|---|---|---|
| mplsNodeIpMapNodeId | This object indicates the Node_ID within the operator. | The tp_get_node_detail function is used to get this object's value. |
| mplsNodeIpMapLocalId | This object contains an IP compatible local identifier that is defined in mplsNodeConfigTable. | The tp_get_node_detail function is used to get this object's value. |
| **mplsTunnelExtTable : The indices of this table are the same as mplsTunnelTable (RFC 3812)** | | |
| mplsTunnelOppositeDirPtr | This object is applicable only for the bidirectional tunnel that has the forward and reverse LSPs in the same tunnel or in different tunnels. This object holds the opposite direction tunnel entry if the bidirectional tunnel is set up by configuring two tunnel entries in mplsTunnelTable.<br><br>The value of zeroDotZero indicates single tunnel entry is used for bidirectional tunnel setup. | Because only one entry per tunnel per LSP for mplsTunnelTable is shown, this object will contain the value 0.0. |
| **mplsTunnelReversePerfTable: This counter is not supported.** | | |
| **mplsNodeIccMapTable:** Because only IP-compatible implementation of the TP is supported, this table is not supported. | | |

# How to Configure MPLS-TP MIB

## Configuring MPLS-TP MIB

A generic SNMP configuration automatically enables MPLS-TP MIB. However, the MPLS TP feature must be configured. See the MPLS Transport Profile document for more information.

You should perform the following generic SNMP configuration tasks:

- Enabling the SNMP agent (required)
- Verifying the status of the SNMP agent (optional)

## Enabling the SNMP Agent

### SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **configure terminal**
4. **snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**][ *number*]

5. **end**
6. **write memory**
7. **show running-config**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| Step 1 | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| Step 2 | **show running-config**<br>**Example:**<br>`Router# show running-config` | Displays the running configuration of the router so that you can determine if an SNMP agent is already running on the device.<br>If no SNMP information is displayed, continue with the next step.<br>If any SNMP information is displayed, you can modify the information or change it as desired. |
| Step 3 | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| Step 4 | **snmp-server community** *string* [**view** *view-name*] [**ro** \| **rw**][ *number*]<br>**Example:**<br>`Router(config)# snmp-server community public ro` | Configures read-only (ro) community strings for the MPLS-TP MIB.<br>• The *string* argument functions like a password, permitting access to SNMP functionality on label switch routers (LSRs) in an MPLS network.<br>• The optional **ro** keyword configures read-only (ro) access to the objects in the MPLS-TP MIB. |
| Step 5 | **end**<br>**Example:**<br>`Router(config)# end` | Exits to privileged EXEC mode. |
| Step 6 | **write memory**<br>**Example:**<br>`Router# write memory` | Writes the modified SNMP configuration into NVRAM of the router, permanently saving the SNMP settings. |
| Step 7 | **show running-config**<br>**Example:**<br>`Router# show running-config` | Displays the running configuration of the router so that you can determine if an SNMP agent is already running on the device. |

| | Command or Action | Purpose |
|---|---|---|
| | | If you see any snmp-server statements, SNMP has been enabled on the router. If any SNMP information is displayed, you can modify the information or change it as desired. |

## 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:

### SUMMARY STEPS

1. **enable**
2. **show running-config**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable** **Example:** `Device> enable` | Enables privileged EXEC mode. • Enter your password if prompted. |
| **Step 2** | **show running-config** **Example:** `Device# show running-config` | Displays the running configuration on the target device. |

# Configuration Examples for MPLS-TP MIB

## Example Enabling the SNMP Agent

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

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

The following example shows how to enable SNMPv1 and SNMPv2C. The configuration permits any SNMP agent to access all MPLS TP MIB objects with read-only permissions using the community string *public*.

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

The following example shows how to allow read-only access to all MPLS TP 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 TP MIB objects.

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

## Example Verifying the Status of the SNMP Agent

The following example shows how to verify the status of the SNMP agent.

```
Device# 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.

## Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| Cisco IOS commands | Cisco IOS Master Commands List, All Releases |
| MPLS commands | Cisco IOS Multiprotocol Label Switching Command Reference |
| MPLS Transport Profile configuration document | MPLS Transport Profile |

### Standards and RFCs

| Standard/RFC | Title |
|---|---|
| draft-ietf-mpls-tp-te-mib-02.txt | MPLS-TP Traffic Engineering (TE) Management Information Base (MIB) |

### Technical Assistance

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

## Feature Information for MPLS-TP MIB

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

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

*Table 1: Feature Information for MPLS-TP MIB*

| Feature Name | Releases | Feature Information |
|---|---|---|
| MPLS-TP MIB | 15.3(1)S<br>XE 3S | Allows you to meet your transport requirements as those requirements evolve from Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) time-division multiplexing (TDM) technologies to MPLS and Ethernet technologies. |