Network Working Group
Internet Draft
Expires: January 2001

Cheenu Srinivasan
Tachion Networks, Inc.

Arun Viswanathan
Force10 Networks, Inc.

Thomas D. Nadeau
Cisco Systems, Inc.
July 2000

MPLS Label Switch Router Management Information Base Using SMIv2

draft-ietf-mpls-lsr-mib-05.txt

Status of this Memo

Table of Contents

1. Abstract

This memo defines an experimental portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects for modeling a Multi-Protocol Label Switching (MPLS) [MPLSArch, MPLSFW] Label Switch Router (LSR).

2. Introduction

This memo defines an experimental portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects for modeling a Multi-Protocol Label Switching (MPLS) [MPLSArch, MPLSFW] Label Switch Router (LSR).

Comments should be made directly to the MPLS mailing list at mpls@uu.net.

This memo does not, in its draft form, specify a standard for the Internet community.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, reference [BCP14].

3. Terminology

This document uses terminology from the document describing the

MPLS architecture [MPLSArch]. A label switched path (LSP) is modeled as a connection consisting of one or more incoming

segments (in-segments) and/or one or more outgoing segments (out-segments) at a label switch router (LSR). The association or interconnection of the in-segments and out-segments is accomplished by using a cross-connect. We use the terminology "connection" and "LSP" interchangeably where the meaning is clear from the context.

4. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

- An overall architecture, described in RFC 2271 [SNMPArch].

- Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and described in RFC 1155 [SMIv1], RFC 1212 [SNMPv1MIBDef] and RFC 1215 [SNMPv1Traps]. The second version, called SMIv2, is described in RFC 1902 [SMIv2], RFC 1903 [SNMPv2TC] and RFC 1904 [SNMPv2Conf].

- Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in RFC 1157 [SNMPv1]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [SNMPv2c] and RFC 1906 [SNMPv2TM]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [SNMPv2TM], RFC 2272 [SNMPv3MP] and RFC 2574 [SNMPv3USM].

- Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in RFC 1157 [SNMPv1]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [SNMPv2PO].

- A set of fundamental applications described in RFC 2273 [SNMPv3App] and the view-based access control mechanism described in RFC 2575 [SNMPv3VACM].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI. This memo specifies a MIB module that is compliant to the SMIv2. A MIB conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically

equivalent, except where objects or events are omitted because no translation is possible (use of Counter64). Some machine-readable

information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine-readable information is not considered to change the semantics of the MIB.

4.1. Object Definitions

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) defined in the SMI. In particular, each object type is named by an OBJECT IDENTIFIER, an administratively assigned name. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the descriptor, to also refer to the object type.

5. Feature Checklist

The MPLS label switch router MIB (LSR-MIB) is designed to satisfy the following requirements and constraints:

- The MIB supports both manually configured LSPs as well as those configured via any MPLS signaling protocol.

- The MIB supports the enabling and disabling of MPLS capability on MPLS capable interfaces of an LSR.

- The MIB allows resource sharing between two or more LSPs.

- Both per-platform and per-interface label spaces are supported.

- MPLS packets can be forwarded solely based on an incoming top label [MPLSArch, LblStk].

- Support is provided for next-hop resolution when the outgoing interface is a shared media interface. In the point-to-multipoint case, each outgoing segment can reside on a different shared media interface.

- The MIB supports point-to-point, point-to-multipoint and multipoint-to-point connections at an LSR.

- For multipoint-to-point connections all outgoing packets can have the same top label.

- For multipoint-to-point connections, the outgoing resources of the merged connections can be shared.

- For multipoint-to-point connections, packets from different incoming connections can have distinct outgoing label stacks beneath the (identical) top label.

- In the point-to-multipoint case each outgoing connection can have a distinct label stack including the top label.

- All the members of a point-to-multipoint connection can share the resources allocated for the ingress segments.

- The MIB provides cross-connect capability to "pop" an incoming label and forward the packet with the remainder of the label stack unchanged and without pushing any labels ("pop-and-go") [LblStk].

- The MIB supports persistent as well as non-persistent LSPs.

- Performance counters are provided for in-segments and out-segments as well as for measuring MPLS performance on a per-interface basis.

6. Outline

Configuring LSPs through an LSR involves the following steps:

- Enabling MPLS on MPLS capable interfaces.

- Configuring in-segments and out-segments.

- Setting up the cross-connect table to associate segments and/or to indicate connection origination and termination.

- Optionally specifying label stack actions.

- Optionally specifying segment traffic parameters.

6.1. Summary of LSR MIB

The MIB objects for performing these actions consist of the following tables:

- The interface configuration table (mplsInterfaceConfTable), which is used for enabling the MPLS protocol on MPLS-capable interfaces.

- The in-segment (mplsInSegmentTable) and out-segment

(mplsOutSegmentTable) tables, which are used for configuring

LSP segments at an LSR.

- The cross-connect table (mplsXCTable), which is used to associate in and out segments together, in order to form a cross-connect.

- The label stack table (mplsLabelStackTable), which is used for specifying label stack operations.

- The Traffic Parameter table (mplsTrafficParamTable), which is used for specifying LSP-related traffic parameters.

Further, the MPLS in-segment and out-segment performance tables, mplsInSegmentPerfTable and mplsOutSegmentPerfTable, contain the objects necessary to measure the performance of LSPs, and mplsInterfacePerfTable has objects to measure MPLS performance on a per-interface basis.

These tables are described in the subsequent sections.

7. Brief Description of MIB Objects

Sections 7.1-7.2 describe objects pertaining to MPLS-capable interfaces of an LSR. The objects described in Sections 7.3-7.8, were derived from the Incoming Label Map (ILM) and Next Hop Label Forwarding Entry (NHLFE) as specified in the MPLS architecture document [MPLSArch]. Section 7.9 describes objects for specifying traffic parameters for in and out segments. It is appropriate to note that the in-segment, out-segment, and cross-connect tables were modeled after similar tables found in [ATOMMIB].

7.1. mplsInterfaceConfTable

This table represents the interfaces that are MPLS capable. An LSR creates an entry in this table for every MPLS capable interface on that LSR.

7.2. mplsInterfacePerfTable

This table contains objects to measure the MPLS performance of MPLS capable interfaces and is an AUGMENT to mplsInterfaceConfTable. High capacity counters are provided for objects that are likely to wrap around quickly on high-speed interfaces.

7.3. mplsInSegmentTable

This table contains a description of the incoming MPLS segments to an LSR and their associated parameters.

The administrative and operational status objects for this table are used to control packet transmission on this segment. If either the administrative and operational status objects for this table are set to 'down', this implies that packets will not be forwarded. Likewise, if the values are set to 'up' this implies that packets are forwarded. These values are particularly useful in cases where multi-point connections utilize a single cross-connect and the administrator wishes to disable some, but not all of the streams. In these cases, the administrator may set the administrative status object to 'down' on some of the in-segments.

7.4. mplsInSegmentPerfTable

The MPLS In-Segment Performance Table has objects to measure the performance of an incoming segment configured on an LSR. It is an AUGMENT to mplsInSegmentTable. High capacity counters are provided for objects that are likely to wrap around quickly on high-speed interfaces.

7.5. mplsOutSegmentTable

The Out-Segment Table contains a description of the outgoing MPLS segments at an LSR and their associated parameters.

The administrative and operational status objects for this table are used to control packet transmission on this segment. If either the administrative and operational status objects are set to 'down', this implies that packets will not be forwarded. Likewise, if the values are set to 'up' this implies that packets are forwarded. These values are particularly useful in cases where multicast connections utilize a single cross-connect and the administrator wishes to disable some, but not all of the streams. In these cases, the administrator may set the administrative status object to 'down' on some of the out-segments.

7.6. mplsOutSegmentPerfTable

The MPLS Out-Segment Table contains objects to measure the performance of an outgoing segment configured on an LSR. It is an AUGMENT to mplsOutSegmentTable. High capacity counters are provided for objects that are likely to wrap around quickly on high-speed interfaces.

7.7. mplsXCTable

The mplsXCTable specifies information for associating segments together in order to instruct the LSR to switch between the specified segments. It supports point-to-point, point-to-multi-point and multi-point-to-point connections.

The administrative and operational status objects for this table imply control of packet forwarding to and from a XCEntry. When the administrative and operational status objects are set to 'down' for example, this implies that the specified XCEntry will not forward packets. Likewise, when either is set to 'up' this implies that packets will be forwarded.

7.8. mplsLabelStackTable

The mplsLabelStackTable specifies the label stack to be pushed onto a packet, beneath the top label. Entries to this table are referred to from mplsXCTable.

7.9. mplsTrafficParamTable

The mplsTrafficParamTable contains objects for specifying the traffic parameters of in-segments and out-segments. Entries in this table are referred to from mplsInSegmentTable and mplsOutSegmentTable.

8. Example of LSP Setup

In this section we provide a brief example of using the MIB objects described in section 11. to set up an LSP. While this example is not meant to illustrate every nuance of the MIB, it is intended as an aid to understanding some of the key concepts. It is meant to be read after going through the MIB itself.

Suppose that one would like to manually create a best-effort, unidirectional LSP. Assume that the LSP enters the LSR via MPLS interface A with ifIndex 12 and exits the LSR via MPLS interface B with ifIndex 13. Let us assume that we do not wish to have a label stack beneath the top label on the outgoing labeled packets. The following example illustrates which rows and corresponding objects might be created to accomplish this.

First, the traffic parameter entries must be set-up for both segments.

In mplsTrafficParamTable for the incoming direction:
{

```
      mplsTrafficParamIndex           = 5
      mplsTrafficParamMaxRate         = 100000,
      mplsTrafficParamMeanRate        = 100000,
      mplsTrafficParamMaxBurstSize    = 2000,
      mplsTrafficParamRowStatus       = createAndGo(4)
   }

   In mplsTrafficParamTable for the outgoing direction:
   {
      mplsTrafficParamIndex           = 6
      mplsTrafficParamMaxRate         = 100000,
      mplsTrafficParamMeanRate        = 100000,
      mplsTrafficParamMaxBurstSize    = 2000,
      mplsTrafficParamRowStatus       = createAndGo(4)
   }
```

Note that if we were setting up a bi-directional LSP, the segments in the reverse direction can share the traffic parameter entries (and hence resources) with the segments in the forward direction.

We must next create the appropriate in-segment and out-segment entries with suitable traffic parameters by pointing to the appropriate traffic parameter entries that we have just created.

```
   In mplsInSegmentTable:
   {
      mplsInSegmentIfIndex            = 12, -- incoming interface
      mplsInSegmentLabel              = 21, -- incoming label
      mplsInSegmentNPop               = 1,
      mplsInSegmentTrafficParamPtr    = mplsTrafficParamIndex.5,
      mplsInSegmentRowStatus          = createAndGo(4)
   }

   In mplsOutSegmentTable:
   {
      mplsOutSegmentIndex             = 1,
      mplsOutSegmentIfIndex           = 13, -- outgoing interface
      mplsOutSegmentPushTopLabel      = true(1),
      mplsOutSegmentTopLabel          = 22, -- outgoing label
      mplsOutSegmentTrafficParamPtr   = mplsTrafficParamIndex.6,
      mplsOutSegmentRowStatus         = createAndGo(4)
   }
```

Next, a cross-connect entry is created thereby associating the newly created segments together.

```
   In mplsXCTable:
```

```
   {
      mplsXCIndex             = 2,
```

```
   mplsXCLspId             = 'c021041502'H, -- 192.33.4.21.2
   mplsInSegmentIfIndex    = 12,
   mplsInSegmentLabel      = 21,
   mplsOutSegmentIndex     = 1,
   mplsXCIsPersistent      = false (1),
   mplsLabelStackIndex     = 0, -- only a single
                                -- outgoing label
   mplsXCRowStatus         = createAndGo(4)
}
```

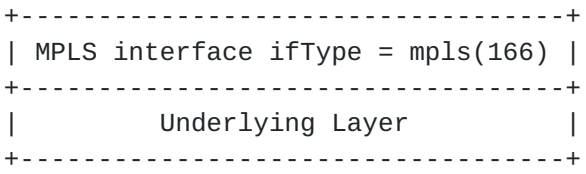
Note that the mplsInSegmentXCIndex and mplsOutSegmentXCIndex objects will automatically be populated with the value 2 when these segments are referred to from the corresponding cross-connect entry.

9. Application of the Interface Group to MPLS

The Interfaces Group of MIB II defines generic managed objects for managing interfaces. This memo contains the media-specific extensions to the Interfaces Group for managing MPLS interfaces.

This memo assumes the interpretation of the Interfaces Group to be in accordance with [IFMIB] which states that the interfaces table (ifTable) contains information on the managed resource's interfaces and that each sub-layer below the internetwork layer of a network interface is considered an interface. Thus, the MPLS interface is represented as an entry in the ifTable. The inter-relation of entries in the ifTable is defined by Interfaces Stack Group defined in [IFMIB].

When using MPLS interfaces, the interface stack table might appear as follows:



In the above diagram, "Underlying Layer" refers to the ifIndex of any interface type, which has been defined for MPLS interworking. Examples include ATM, Frame Relay, Ethernet, etc.

9.1. Support of the MPLS Layer by ifTable

Some specific interpretations of ifTable for the MPLS layer

follow.

Object	Use for the MPLS layer
ifIndex	Each MPLS interface is represented by an ifEntry.
ifDescr	Description of the MPLS interface.
ifType	The value that is allocated for MPLS is 166.
ifSpeed	The total bandwidth in bits per second for use by the MPLS layer.
ifPhysAddress	Unused.
ifAdminStatus	This variable indicates the administrator's intent as to whether MPLS should be enabled, disabled, or running in some diagnostic testing mode on this interface. Also see [IFMIB].
ifOperStatus	This value reflects the actual or operational status of MPLS on this interface.
ifLastChange	See [IFMIB].
ifInOctets	The number of received octets over the interface, i.e., the number of received, octets received as labeled packets.
ifOutOctets	The number of transmitted octets over the interface, i.e., the number of octets transmitted as labeled packets.
ifInErrors	The number of labeled packets dropped due to uncorrectable errors.
ifInUnknownProtos	The number of received packets discarded during packet header validation, including packets with unrecognized label values.
ifOutErrors	See [IFMIB].
ifName	Textual name (unique on this system) of the interface or an octet string of zero length.
ifLinkUpDownTrapEnable	Default is disabled (2).
ifConnectorPresent	

Set to false (2).

ifHighSpeed See [IFMIB].

ifHCInOctets The 64-bit version of ifInOctets; supported if required by the compliance statements in [IFMIB].

ifHCOutOctets The 64-bit version of ifOutOctets; supported if required by the compliance statements in [IFMIB].

ifAlias The non-volatile 'alias' name for the interface as specified by a network manager.

ifCounterDiscontinuityTime
See [IFMIB].

10. The Use of RowPointer

RowPointer is a textual convention used to identify a conceptual row in an SNMP Table by pointing to one of its objects. In this MIB, it is used in mplsInSegmentTable and mplsOutSegmentTable for the following purposes. First, it indicates a particular traffic parameter table. An example of such a table is mplsTrafficParamTable. Second, it is used to indicate a specific instance of a traffic parameter entry that is associated with a given in-segment or out-segment entry. In the in-segment and out-segment tables, the trafficParamPtr SHOULD point to the first column of the appropriate conceptual row.

11. MPLS Label Switch Router MIB Definitions

```
MPLS-LSR-MIB DEFINITIONS ::= BEGIN

IMPORTS
   MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
   experimental, Integer32, Counter32, Unsigned32,
   Counter64, Gauge32
      FROM SNMPv2-SMI

   MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
      FROM SNMPv2-CONF

   TEXTUAL-CONVENTION, TruthValue, RowStatus, StorageType,
   RowPointer, TimeStamp
      FROM SNMPv2-TC

   InterfaceIndex, InterfaceIndexOrZero
      FROM IF-MIB
```

AddressFamilyNumbers

```
      FROM IANA-ADDRESS-FAMILY-NUMBERS-MIB

   InetAddressIPv4, InetAddressIPv6, InetAddressType
      FROM INET-ADDRESS-MIB;


mplsLsrMIB MODULE-IDENTITY
   LAST-UPDATED "200007071200Z"  -- 07 July 2000 12:00:00 EST
   ORGANIZATION "Multiprotocol Label Switching (MPLS) Working Group"
   CONTACT-INFO
       "        Cheenu Srinivasan
        Postal: Tachion Networks, Inc.
                Monmouth Park Corporate Center I
                Building C, 185 Monmouth Parkway
                West Long Branch, NJ 07764
        Tel:    +1-732-542-7750 x1234
        Email:  cheenu@tachion.com

                Arun Viswanathan
        Postal: Force10 Networks, Inc.
                1440 McCarthy Blvd
                Milpitas, CA 95035
        Tel:    +1-408-571-3516
        Email:  arun@force10networks.com

                Thomas D. Nadeau
        Postal: Cisco Systems, Inc.
                250 Apollo Drive
                Chelmsford, MA 01824
        Tel:    +1-978-244-3051
        Email:  tnadeau@cisco.com"

   DESCRIPTION
       "This MIB contains managed object definitions for the
        Multiprotocol Label Switching (MPLS) Router as
        defined in: Rosen, E., Viswanathan, A., and R.
        Callon, Multiprotocol Label Switching Architecture,
        Internet Draft <draft-ietf-mpls-arch-06.txt>,
        August 1999."

   -- Revision history.
   REVISION
       "200007071200Z"  -- 07 July 2000 12:00:00 EST
   DESCRIPTION
       "Sixth draft version. Made minor typographical corrections
        noted from WG mailing list during second working group last
        call."
   REVISION
```

```
        "200004261200Z"  -- 26 April 2000 12:00:00 EST
    DESCRIPTION
```

```
        "Fifth draft version. Made minor typographical corrections
         noted from WG mailing list."
    REVISION
        "200004211200Z"  -- 21 April 2000 12:00:00 EST
    DESCRIPTION
        "Fourth draft version.  Made corrections from WG Last
         Call comments."
    REVISION
        "200003061200Z"  -- 6 March 2000 12:00:00 EST
    DESCRIPTION
        "Third draft version."
    REVISION
        "200002161200Z"  -- 16 February 2000 12:00:00 EST
    DESCRIPTION
        "Second draft version."
    REVISION
        "199906161200Z"  -- 16 June 1999 12:00:00 EST
    DESCRIPTION
       "Initial draft version."
    ::= { experimental 96 }


-- Textual Conventions.

MplsLSPID ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
        "An identifier that is assigned to each LSP and is
         used to uniquely identify it.  This is assigned at
         the head end of the LSP and can be used by all LSRs
         to identify this LSP.  This value is piggybacked by
         the signaling protocol when this LSP is signaled
         within the network.  This identifier can then be
         used at each LSR to identify which labels are being
         swapped to other labels for this LSP.  For IPv4
         addresses this results in a 6-octet long cookie."
    SYNTAX        OCTET STRING (SIZE (0..31))

-- An MPLS label.
MplsLabel ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
        "This value represents an MPLS label.  Note that the
         contents of a label field are interpreted in an
         interface-type specific fashion.  For example, the
         20-bit wide label carried in the MPLS shim header is
         contained in bits 0-19 and bits 20-31 must be zero.
```

The frame relay label can be either 10 or 23 bits
wide depending on the size of the DLCI field and
bits 10-31, or 23-31 must be zero, respectively.

```
        For an ATM interface, bits 0-15 must be interpreted
        as the VCI, bits 16-23 as the VPI and bits 24-31
        must be zero.  Note that the permissible label
        values are also a function of the interface type.
        For example, the value 3 has special semantics in
        the control plane for an MPLS shim header label and
        is not a valid label value in the data path."

   REFERENCE
       "1. MPLS Label Stack Encoding, Rosen et al, draft-
           ietf-mpls-label-encaps-07.txt, March 2000.
        2. Use of Label Switching on Frame Relay Networks,
           Conta et al, draft-ietf-mpls-fr-03.txt, Nov.
           1998.
        3. MPLS using LDP and ATM VC switching, Davie et al,
           draft-ietf-mpls-atm-02.txt, April 1999."
   SYNTAX  Unsigned32 (0..4294967295)


MplsBitRate ::= TEXTUAL-CONVENTION
   DISPLAY-HINT "d"
   STATUS      current
   DESCRIPTION
       "An estimate of bandwidth in units of 1,000 bits per
        second.  If this object reports a value of 'n' then
        the rate of the object is somewhere in the range of
        'n-500' to 'n+499'. For objects which do not vary in
        bitrate, or for those where no accurate estimation
        can be made, this object should contain the nominal
        bitrate."
   SYNTAX  Integer32 (1..2147483647)


MplsBurstSize ::= TEXTUAL-CONVENTION
   DISPLAY-HINT "d"
   STATUS      current
   DESCRIPTION
       "The number of octets of MPLS data that the stream
        may send back-to-back without concern for policing."
   SYNTAX  Integer32 (1..2147483647)



MplsObjectOwner ::= TEXTUAL-CONVENTION
   STATUS      current
   DESCRIPTION
           "The entity which owns the object in question."
   SYNTAX   INTEGER {
```

```
                other(1),
                management(2),
                ldp(3),
                rsvp(4),
                crldp(5),
                policyAgent(6),
                unknown (7)
             }

-- Top level components of this MIB.

-- Tables, Scalars
mplsLsrObjects       OBJECT IDENTIFIER ::= { mplsLsrMIB 1 }
-- traps
mplsLsrNotifications OBJECT IDENTIFIER ::= { mplsLsrMIB 2 }
mplsLsrNotifyPrefix OBJECT IDENTIFIER ::= { mplsLsrNotifications 0 }
-- conformance
mplsLsrConformance   OBJECT IDENTIFIER ::= { mplsLsrMIB 3 }


-- MPLS Interface Configuration Table.

mplsInterfaceConfTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsInterfaceConfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table specifies per-interface MPLS capability
        and associated information."
   ::= { mplsLsrObjects 1 }

mplsInterfaceConfEntry OBJECT-TYPE
   SYNTAX        MplsInterfaceConfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table is created by an LSR for
        every interface capable of supporting MPLS. The
        entry with index 0 represents the per-platform label
        space and contains parameters that apply to all
        interfaces that participate in the per-platform
        label space. Other entries defined in this table
        represent additional MPLS interfaces that may
        participate in either the per-platform or per-
        interface label spaces, or both. Additional
        information about label space participation of an
        interface is provided in the description clause of
```

```
        mplsInterfaceLabelParticipationType."
    INDEX       { mplsInterfaceConfIndex }
       ::= { mplsInterfaceConfTable 1 }
```

```
MplsInterfaceConfEntry ::= SEQUENCE {
  mplsInterfaceConfIndex              InterfaceIndexOrZero,
  mplsInterfaceLabelMinIn             MplsLabel,
  mplsInterfaceLabelMaxIn             MplsLabel,
  mplsInterfaceLabelMinOut            MplsLabel,
  mplsInterfaceLabelMaxOut            MplsLabel,
  mplsInterfaceTotalBandwidth         MplsBitRate,
  mplsInterfaceAvailableBandwidth     MplsBitRate,
  mplsInterfaceLabelParticipationType BITS,
  mplsInterfaceConfStorageType        StorageType
}

mplsInterfaceConfIndex OBJECT-TYPE
   SYNTAX        InterfaceIndexOrZero
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This is a unique index for an entry in the
        MplsInterfaceConfTable.  A non-zero index for an
        entry indicates the ifIndex for the corresponding
        interface entry in of the MPLS-layer in the ifTable.
        Note that the per-platform label space may apply to
        several interfaces, and therefore the configuration
        of the per-platform label space interface parameters
        will apply to all of the interfaces that are
        participating in the per-platform label space."
   REFERENCE
       "RFC 2233 - The Interfaces Group MIB using SMIv2,
        McCloghrie, K., and F. Kastenholtz, Nov. 1997"
   ::= { mplsInterfaceConfEntry 1 }

mplsInterfaceLabelMinIn OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This is the minimum value of an MPLS label that this
        LSR is willing to receive on this interface."
   ::= { mplsInterfaceConfEntry 2 }

mplsInterfaceLabelMaxIn OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This is the maximum value of an MPLS label that this
```

```
        LSR is willing to receive on this interface."
    ::= { mplsInterfaceConfEntry 3 }
```

```
mplsInterfaceLabelMinOut OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This is the minimum value of an MPLS label that this
        LSR is willing to send on this interface."
   ::= { mplsInterfaceConfEntry 4 }

mplsInterfaceLabelMaxOut OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This is the maximum value of an MPLS label that this
        LSR is willing to send on this interface."
   ::= { mplsInterfaceConfEntry 5 }

mplsInterfaceTotalBandwidth        OBJECT-TYPE
   SYNTAX        MplsBitRate
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This value indicates the total amount of usable
        bandwidth on this interface and is specified in
        kilobits per second (Kbps).  This variable is
        not applicable when applied to the interface with
        index 0."
::= { mplsInterfaceConfEntry 6 }

mplsInterfaceAvailableBandwidth       OBJECT-TYPE
   SYNTAX        MplsBitRate
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This value indicates the total amount of available
        bandwidth available on this interface and is
        specified in kilobits per second (Kbps).  This
        value is calculated as the difference between the
        amount of bandwidth currently in use and that
        specified in mplsInterfaceTotalBandwidth.  This
        variable is not applicable when applied to the
        interface with index 0."
::= { mplsInterfaceConfEntry 7 }

mplsInterfaceLabelParticipationType OBJECT-TYPE
   SYNTAX  BITS {
```

```
            perPlatform (0),
            perInterface (1)
          }
```

```
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "Either the perPlatform(0) or perInterface(1) bit
        MUST be set. If the value of the
        mplsInterfaceConfIndex for this entry is zero, then
        only the perPlatform(0) bit MUST be set and the
        perInterface(1) bit is meaningless. If the
        perInterface(1) bit is set then the value of
        mplsInterfaceLabelMinIn, mplsInterfaceLabelMaxIn,
        mplsInterfaceLabelMinOut, and
        mplsInterfaceLabelMaxOut for this entry reflect the
        label ranges for this interface.  If only the
        perPlatform(0) bit is set, then the value of
        mplsInterfaceLabelMinIn, mplsInterfaceLabelMaxIn,
        mplsInterfaceLabelMinOut, and
        mplsInterfaceLabelMaxOut for this entry must be
        identical to the instance of these objects with
        index 0."
   REFERENCE
      "Multiprotocol Label Switching, Rosen et al, draft-ietf-mpls-
       arch-06.txt, August 1999."
::= { mplsInterfaceConfEntry 8 }

mplsInterfaceConfStorageType  OBJECT-TYPE
   SYNTAX      StorageType
   MAX-ACCESS  read-create
   STATUS      current
   DESCRIPTION
       "The storage type for this entry."
   ::= { mplsInterfaceConfEntry 9 }

-- End of mplsInterfaceConfTable


-- MPLS Interface Performance Table.
mplsInterfacePerfTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsInterfacePerfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table provides MPLS performance information on
        a per-interface basis."
   ::= { mplsLsrObjects 2 }

mplsInterfacePerfEntry OBJECT-TYPE
   SYNTAX        MplsInterfacePerfEntry
```

```
    MAX-ACCESS    not-accessible
    STATUS        current
    DESCRIPTION
```

```
       "An entry in this table is created by the LSR for
        every interface capable of supporting MPLS.  Its is
        an extension to the mplsInterfaceConfEntry table."
   AUGMENTS      { mplsInterfaceConfEntry }
      ::= { mplsInterfacePerfTable 1 }

MplsInterfacePerfEntry ::= SEQUENCE {
      -- incoming direction
      mplsInterfaceInLabelsUsed          Gauge32,
      mplsInterfaceFailedLabelLookup     Counter32,

      -- outgoing direction
      mplsInterfaceOutLabelsUsed         Gauge32,
      mplsInterfaceOutFragments          Counter32
   }

mplsInterfaceInLabelsUsed OBJECT-TYPE
   SYNTAX        Gauge32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object counts the number of labels
        that are in use at this point in time on this
        interface in the incoming direction. If the interface
        participates in the per-platform label space only,
        then this instance of this object MUST be identical
        with the instance with index 0. If the interface
        participates in the per-interface label space, then this
        this instance of this object MUST represent the number of
        of per-interface labels that are in use at this point in
        time on this interface."
   ::= { mplsInterfacePerfEntry 1 }

mplsInterfaceFailedLabelLookup OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object counts the number of labeled packets
        that have been received on this interface and were
        discarded because there was no matching cross-connect
        entry. This object MUST count on a per-interface basis
        regardless of which label space the interface participates
        in."
   ::= { mplsInterfacePerfEntry 2 }

mplsInterfaceOutLabelsUsed OBJECT-TYPE
```

```
   SYNTAX        Gauge32
   MAX-ACCESS    read-only
   STATUS        current
```

```
   DESCRIPTION
       "This object counts the number of top-most labels in the
        outgoing label stacks that are in use at this point
        in time on this interface. This object
        MUST count on a per-interface basis regardless of
        which label space the interface participates in."
   ::= { mplsInterfacePerfEntry 3 }

mplsInterfaceOutFragments OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object counts the number of outgoing MPLS
        packets that required fragmentation before
        transmission on this interface. This object
        transmission on this interface. This object
        MUST count on a per-interface basis regardless of
        which label space the interface participates in."
::= { mplsInterfacePerfEntry 4 }

-- In-segment table.

mplsInSegmentTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsInSegmentEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table contains a collection of incoming
        segments to an LSR."
   ::= { mplsLsrObjects 3 }

mplsInSegmentEntry  OBJECT-TYPE
   SYNTAX        MplsInSegmentEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table represents one incoming
        segment.  An entry can be created by a network
        administrator or an SNMP agent, or an MPLS signaling
        protocol.  The creator of the entry is denoted by
        mplsInSegmentOwner.  An entry in this table is
        indexed by the ifIndex of the incoming interface and
        the (top) label."
   INDEX         { mplsInSegmentIfIndex, mplsInSegmentLabel }
   ::= { mplsInSegmentTable 1 }
```

```
MplsInSegmentEntry ::= SEQUENCE {
  mplsInSegmentIfIndex              InterfaceIndexOrZero,
  mplsInSegmentLabel                MplsLabel,
```

```
  mplsInSegmentNPop                 Integer32,
  mplsInSegmentAddrFamily           AddressFamilyNumbers,
  mplsInSegmentXCIndex              Integer32,
  mplsInSegmentOwner                MplsObjectOwner,
  mplsInSegmentTrafficParamPtr      RowPointer,
  mplsInSegmentRowStatus            RowStatus,
  mplsInSegmentStorageType          StorageType
}

mplsInSegmentIfIndex OBJECT-TYPE
   SYNTAX        InterfaceIndexOrZero
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This is a unique index for an entry in the
        MplsInSegmentTable.  This value represents the
        interface index for the incoming MPLS interface.  A
        value of zero represents an incoming label from the
        per-platform label space.  In this case, the
        mplsInSegmentLabel is interpreted to be an MPLS-type
        label."
   ::= { mplsInSegmentEntry 1 }

mplsInSegmentLabel OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "The incoming label for this segment."
   ::= { mplsInSegmentEntry 2 }

mplsInSegmentNPop OBJECT-TYPE
   SYNTAX        Integer32 (1..2147483647)
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "The number of labels to pop from the incoming
        packet.  Normally only the top label is popped from
        the packet and used for all switching decisions for
        that packet.  Note that technologies which do not
        support label popping should set this value to its
        default value of 1."
   DEFVAL        { 1 }
   ::= { mplsInSegmentEntry 3 }

mplsInSegmentAddrFamily OBJECT-TYPE
   SYNTAX        AddressFamilyNumbers
```

```
    MAX-ACCESS    read-create
    STATUS        current
    DESCRIPTION
```

```
       "The IANA address family [IANAFamily] of the incoming
        packet.  A value of other(0) indicates that the
        family type is either unknown or undefined."
   DEFVAL        { other }
   ::= { mplsInSegmentEntry 4 }

mplsInSegmentXCIndex OBJECT-TYPE
SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "Index into mplsXCTable which identifies which cross-
        connect entry this segment is part of.  A value of
        zero indicates that this entry is not referred to by
        any cross-connect entry.  When a cross-connect entry
        is created which this in-segment is a part of, this
        object is automatically updated to reflect the value
        of mplsXCIndex of that cross-connect entry."
   DEFVAL        { 0 }
   ::= { mplsInSegmentEntry 5 }

mplsInSegmentOwner OBJECT-TYPE
   SYNTAX        MplsObjectOwner
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Denotes the entity that created and is responsible
        for managing this segment."
   DEFVAL { unknown }
   ::= { mplsInSegmentEntry 6 }

mplsInSegmentTrafficParamPtr OBJECT-TYPE
   SYNTAX           RowPointer
   MAX-ACCESS       read-create
   STATUS           current
   DESCRIPTION
     "This variable represents a pointer to the traffic
     parameter specification for this in-segment.  This
     value may point at an entry in the
     mplsTrafficParamTable to indicate which
     mplsTrafficParamEntry is to be assigned to this
     segment.  This value may optionally point at an
     externally defined traffic parameter specification
     table.  A value of zero-dot-zero indicates best-effort
     treatment.  By having the same value of this object,
     two or more segments can indicate resource sharing."
   ::= { mplsInSegmentEntry 7 }
```

```
mplsInSegmentRowStatus OBJECT-TYPE
   SYNTAX        RowStatus
```

```
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "This variable is used to create, modify, and/or
        delete a row in this table."
   ::= { mplsInSegmentEntry 8 }

mplsInSegmentStorageType OBJECT-TYPE
   SYNTAX        StorageType
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "This variable indicates the storage type for this
        object."
   ::= { mplsInSegmentEntry 9 }

-- End of mplsInSegmentTable

-- In-segment performance table.

mplsInSegmentPerfTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsInSegmentPerfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table contains statistical information for
        incoming MPLS segments to an LSR."
   ::= { mplsLsrObjects 4 }

mplsInSegmentPerfEntry  OBJECT-TYPE
   SYNTAX        MplsInSegmentPerfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table contains statistical
        information about one incoming segment which was
        configured in the mplsInSegmentTable. The counters
        in this entry should behave in a manner similar to
        that of the interface."
   AUGMENTS      { mplsInSegmentEntry }
   ::= { mplsInSegmentPerfTable 1 }

MplsInSegmentPerfEntry ::= SEQUENCE {
    mplsInSegmentOctets                Counter32,
    mplsInSegmentPackets               Counter32,
    mplsInSegmentErrors                Counter32,
    mplsInSegmentDiscards              Counter32,
```

```
    -- high capacity counter
    mplsInSegmentHCOctets               Counter64,
```

```
    mplsInSegmentPerfDiscontinuityTime  TimeStamp
   }

mplsInSegmentOctets OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This value represents the total number of octets
        received by this segment."
   ::= { mplsInSegmentPerfEntry 1 }

mplsInSegmentPackets OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "Total number of packets received by this segment."
   ::= { mplsInSegmentPerfEntry 2 }

mplsInSegmentErrors OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "The number of errored packets received on this
        segment."
   ::= { mplsInSegmentPerfEntry 3 }

mplsInSegmentDiscards OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "The number of labeled packets received on this in-
        segment, which were chosen to be discarded even
        though no errors had been detected to prevent their
        being transmitted.  One possible reason for
        discarding such a labeled packet could be to free up
        buffer space."
   ::= { mplsInSegmentPerfEntry 4 }

mplsInSegmentHCOctets OBJECT-TYPE
   SYNTAX        Counter64
   MAX-ACCESS    read-only
   STATUS        current
```

```
    DESCRIPTION
        "The total number of octets received.  This is the 64
         bit version of mplsInSegmentOctets."
```

```
   ::= { mplsInSegmentPerfEntry 5 }

mplsInSegmentPerfDiscontinuityTime OBJECT-TYPE
       SYNTAX      TimeStamp
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
         "The value of sysUpTime on the most recent occasion at
          which any one or more of this segment's Counter32 or
          Counter64 suffered a discontinuity. If no such
          discontinuities have occurred since the last re-
          initialization of the local management subsystem, then
          this object contains a zero value."
       ::= { mplsInSegmentPerfEntry 6 }


-- End of mplsInSegmentPerfTable.

-- Out-segment table.

mplsOutSegmentIndexNext OBJECT-TYPE
   SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object contains the next appropriate value to
        be used for mplsOutSegmentIndex when creating
        entries in the mplsOutSegmentTable.  If the number
        of unassigned entries is exhausted, this object will
        take on the value of 0.  To obtain the
        mplsOutSegmentIndex value for a new entry, the
        manager must first issue a management protocol
        retrieval operation to obtain the current value of
        this object.  The agent should modify the value to
        reflect the next unassigned index after each
        retrieval operation.  After a manager retrieves a
        value the agent will determine through its local
        policy when this index value will be made available
        for reuse."
   ::= { mplsLsrObjects 5 }


mplsOutSegmentTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsOutSegmentEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table contains a representation of the outgoing
```

```
        segments from an LSR."
    ::= { mplsLsrObjects 6 }
```

```
mplsOutSegmentEntry  OBJECT-TYPE
   SYNTAX        MplsOutSegmentEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table represents one outgoing
        segment.  An entry can be created by a network
        administrator or an SNMP agent, or an MPLS signaling
        protocol.  The object mplsOutSegmentOwner indicates
        the creator of this entry."
   INDEX         { mplsOutSegmentIndex }
   ::= { mplsOutSegmentTable 1 }

MplsOutSegmentEntry ::= SEQUENCE {
  mplsOutSegmentIndex               Integer32,
  mplsOutSegmentIfIndex             InterfaceIndex,
  mplsOutSegmentPushTopLabel        TruthValue,
  mplsOutSegmentTopLabel            MplsLabel,
  mplsOutSegmentNextHopIpAddrType   InetAddressType,
  mplsOutSegmentNextHopIpv4Addr     InetAddressIPv4,
  mplsOutSegmentNextHopIpv6Addr     InetAddressIPv6,
  mplsOutSegmentXCIndex             Integer32,
  mplsOutSegmentOwner               MplsObjectOwner,
  mplsOutSegmentTrafficParamPtr     RowPointer,
  mplsOutSegmentRowStatus           RowStatus,
  mplsOutSegmentStorageType         StorageType
}

mplsOutSegmentIndex OBJECT-TYPE
   SYNTAX        Integer32(0..2147483647)
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This value contains a unique index for this row.
        While a value of 0 is not valid as an index for this
        row it can be supplied as a valid value to index
        mplsXCTable to access entries for which no out-
        segment has been configured."
   ::= { mplsOutSegmentEntry 1 }

mplsOutSegmentIfIndex OBJECT-TYPE
   SYNTAX        InterfaceIndex
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "This value contains the interface index of the
```

```
        outgoing interface."
    ::= { mplsOutSegmentEntry 2 }
```

```
mplsOutSegmentPushTopLabel OBJECT-TYPE
   SYNTAX        TruthValue
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "This value indicates whether or not a top label
        should be pushed onto the outgoing packet's label
        stack.  The value of this variable must be set to
        true if the outgoing interface does not support pop-
        and-go (for example an ATM interface) or if it is a
        tunnel origination.  Note that it is considered an
        error in the case that mplsOutSegmentPushTopLabel is
        set to false, but the cross-connect entry which
        refers to this out-segment has a non-zero
        mplsLabelStackIndex.  The LSR MUST ensure that this
        situation does not happen "
   ::= { mplsOutSegmentEntry 3 }

mplsOutSegmentTopLabel OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "If mplsOutSegmentPushTopLabel is true then this
        represents the label that should be pushed onto the
        top of the outgoing packet's label stack."
   ::= { mplsOutSegmentEntry 4 }

mplsOutSegmentNextHopIpAddrType OBJECT-TYPE
   SYNTAX        InetAddressType
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Indicates whether the next hop address is IPv4 or
        IPv6.  Note that a value of unknown (0) is valid
        only when the outgoing interface is of type point-to-
        point."
   DEFVAL        { unknown }
   ::= { mplsOutSegmentEntry 5 }

mplsOutSegmentNextHopIpv4Addr OBJECT-TYPE
   SYNTAX        InetAddressIPv4
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "IPv4 Address of the next hop.  Its value is
        significant only when
```

```
        mplsOutSegmentNextHopIpAddrType is ipV4 (1),
        otherwise it should return a value of 0."
    ::= { mplsOutSegmentEntry 6 }
```

```
mplsOutSegmentNextHopIpv6Addr OBJECT-TYPE
   SYNTAX        InetAddressIPv6
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "IPv6 address of the next hop.  Its value is
        significant only when
        mplsOutSegmentNextHopIpAddrType is ipV6 (2),
        otherwise it should return a value of 0."
   ::= { mplsOutSegmentEntry 7 }

mplsOutSegmentXCIndex OBJECT-TYPE
   SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "Index into mplsXCTable which identifies which cross-
        connect entry this segment is part of.  A value of
        zero indicates that this entry is not referred to by
        any cross-connect entry.  When a cross-connect entry
        is created which this out-segment is a part of, this
        object is automatically updated to reflect the value
        of mplsXCIndex of that cross-connect entry."
   DEFVAL        { 0 }
   ::= { mplsOutSegmentEntry 8 }

mplsOutSegmentOwner OBJECT-TYPE
   SYNTAX        MplsObjectOwner
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Denotes the entity which created and is responsible
        for managing this segment."
   DEFVAL { unknown }
   ::= { mplsOutSegmentEntry 9 }

mplsOutSegmentTrafficParamPtr OBJECT-TYPE
   SYNTAX        RowPointer
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
     "This variable represents a pointer to the traffic
      parameter specification for this out-segment.  This
      value may point at an entry in the
      mplsTrafficParamTable to indicate which
      mplsTrafficParamEntry is to be assigned to this
```

segment. This value may optionally point at an
externally defined traffic parameter specification
table. A value of zero-dot-zero indicates best-

```
      effort treatment.  By having the same value of this
      object, two or more segments can indicate resource
      sharing."
   ::= { mplsOutSegmentEntry 10 }

mplsOutSegmentRowStatus OBJECT-TYPE
   SYNTAX        RowStatus
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "For creating, modifying, and deleting this row."
   ::= { mplsOutSegmentEntry 11 }

mplsOutSegmentStorageType OBJECT-TYPE
   SYNTAX        StorageType
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "This variable indicates the storage type for this
        object."
   ::= { mplsOutSegmentEntry 12 }

-- End of mplsOutSegmentTable


-- Out-segment performance table.

mplsOutSegmentPerfTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsOutSegmentPerfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table contains statistical information about
        outgoing segments from an LSR. The counters in this
        entry should behave in a manner similar to that of
        the interface."
   ::= { mplsLsrObjects 7 }

mplsOutSegmentPerfEntry  OBJECT-TYPE
   SYNTAX        MplsOutSegmentPerfEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table contains statistical
        information about one outgoing segment configured in
        mplsOutSegmentTable."
   AUGMENTS      { mplsOutSegmentEntry }
      ::= { mplsOutSegmentPerfTable 1 }
```

```
MplsOutSegmentPerfEntry ::= SEQUENCE {
```

```
      mplsOutSegmentOctets                Counter32,
      mplsOutSegmentPackets               Counter32,
      mplsOutSegmentErrors                Counter32,
      mplsOutSegmentDiscards              Counter32,

      -- HC counter
      mplsOutSegmentHCOctets              Counter64,

      mplsOutSegmentPerfDiscontinuityTime  TimeStamp
   }

mplsOutSegmentOctets OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This value contains the total number of octets sent
        on this segment."
   ::= { mplsOutSegmentPerfEntry 1 }

mplsOutSegmentPackets OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This value contains the total number of packets sent
        on this segment."
   ::= { mplsOutSegmentPerfEntry 2 }

mplsOutSegmentErrors OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "Number of packets that could not be sent due to
        errors on this segment."
   ::= { mplsOutSegmentPerfEntry 3 }

mplsOutSegmentDiscards OBJECT-TYPE
   SYNTAX        Counter32
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "The number of labeled packets received on this out-
        segment, which were chosen to be discarded even
        though no errors had been detected to prevent their
        being transmitted.  One possible reason for
```

```
        discarding such a labeled packet could be to free up
        buffer space."
    ::= { mplsOutSegmentPerfEntry 4 }
```

```
mplsOutSegmentHCOctets OBJECT-TYPE
   SYNTAX        Counter64
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "Total number of octets sent.  This is the 64 bit
        version of mplsOutSegmentOctets."
   ::= { mplsOutSegmentPerfEntry 5 }

mplsOutSegmentPerfDiscontinuityTime OBJECT-TYPE
       SYNTAX      TimeStamp
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
         "The value of sysUpTime on the most recent occasion at
          which any one or more of this segment's Counter32 or
          Counter64 suffered a discontinuity. If no such
          discontinuities have occurred since the last re-
          initialization of the local management subsystem, then
          this object contains a zero value."
       ::= { mplsOutSegmentPerfEntry 6 }


-- End of mplsOutSegmentPerfTable.

-- Cross-connect table.

mplsXCIndexNext OBJECT-TYPE
   SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object contains an appropriate value to be used
        for mplsXCIndex when creating entries in the
        mplsXCTable.  The value 0 indicates that no
        unassigned entries are available.  To obtain the
        value of mplsXCIndex for a new entry in the
        mplsXCTable, the manager issues a management
        protocol retrieval operation to obtain the current
        value of mplsXCIndex.  After each retrieval
        operation, the agent should modify the value to
        reflect the next unassigned index.  After a manager
        retrieves a value the agent will determine through
        its local policy when this index value will be made
        available for reuse."
   ::= { mplsLsrObjects 8 }
```

```
mplsXCTable  OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsXCEntry
```

```
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table specifies information for switching
        between LSP segments.  It supports point-to-point,
        point-to-multipoint and multipoint-to-point
        connections.  mplsLabelStackTable specifies the
        label stack information for a cross-connect LSR and
        is referred to from mplsXCTable."
   ::= { mplsLsrObjects 9 }

mplsXCEntry  OBJECT-TYPE
   SYNTAX        MplsXCEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "A row in this table represents one cross-connect
        entry.  The following objects index it:

        - cross-connect index mplsXCIndex that uniquely
          identifies a group of cross-connect entries
        - interface index of the in-segment,
          mplsInSegmentIfIndex
        - incoming label(s), mplsInSegmentLabel
        - out-segment index, mplsOutSegmentIndex

       Originating LSPs:
        These are represented by using the special
        combination of values mplsInSegmentIfIndex=0 and
        mplsInSegmentLabel=0 as indexes.  In this case the
        mplsOutSegmentIndex MUST be non-zero.

       Terminating LSPs:
       These are represented by using the special value
        mplsOutSegmentIndex=0 as index.

       Special labels:
        Entries indexed by reserved MPLS label values 0
        through 15 imply terminating LSPs and MUST have
        mplsOutSegmentIfIndex = 0.  Note that situations
        where LSPs are terminated with incoming label equal
        to 0, should have mplsInSegmentIfIndex = 0 as well,
        but can be distinguished from originating LSPs
        because the mplsOutSegmentIfIndex = 0.  The
        mplsOutSegmentIfIndex MUST only be set to 0 in
        cases of terminating LSPs.
```

```
        An entry can be created by a network administrator
        or by an SNMP agent as instructed by an MPLS
        signaling protocol."
```

```
   INDEX  { mplsXCIndex, mplsInSegmentIfIndex,
            mplsInSegmentLabel,
            mplsOutSegmentIndex }
      ::= { mplsXCTable 1 }

MplsXCEntry ::= SEQUENCE {
      mplsXCIndex                Integer32,
      mplsXCLspId                MplsLSPID,
      mplsXCLabelStackIndex      Integer32,
      mplsXCIsPersistent         TruthValue,
      mplsXCOwner                MplsObjectOwner,
      mplsXCRowStatus            RowStatus,
      mplsXCStorageType          StorageType,
      mplsXCAdminStatus          INTEGER,
      mplsXCOperStatus           INTEGER
   }

mplsXCIndex OBJECT-TYPE
   SYNTAX        Integer32 (1..2147483647)
   MAX-ACCESS    accessible-for-notify
   STATUS        current
   DESCRIPTION
       "Primary index for the conceptual row identifying
        a group of cross-connect segments."
   ::= { mplsXCEntry 1 }

mplsXCLspId OBJECT-TYPE
   SYNTAX        MplsLSPID
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "This value identifies the label switched path that
        this cross-connect entry belongs to."
   ::= { mplsXCEntry 2 }

mplsXCLabelStackIndex OBJECT-TYPE
   SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Primary index into mplsLabelStackTable identifying a
        stack of labels to be pushed beneath the top label.
        Note that the top label identified by the out-
        segment ensures that all the components of a
        multipoint-to-point connection have the same
        outgoing label.  A value of 0 indicates that no
        labels are to be stacked beneath the top label."
```

```
   ::= { mplsXCEntry 3 }

mplsXCIsPersistent OBJECT-TYPE
```

```
   SYNTAX        TruthValue
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Denotes whether or not this cross-connect entry and
        associated in- and out-segments should be restored
        automatically after failures.  This value MUST be
        set to false in cases where this cross-connect entry
        was created by a signaling protocol."
   DEFVAL        { false }
   ::= { mplsXCEntry 4 }

mplsXCOwner OBJECT-TYPE
   SYNTAX    MplsObjectOwner
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Denotes the entity that created and is responsible
        for managing this cross-connect."
   ::= { mplsXCEntry 5 }

mplsXCRowStatus OBJECT-TYPE
   SYNTAX        RowStatus
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "For creating, modifying, and deleting this row."
   ::= { mplsXCEntry 6 }

mplsXCStorageType OBJECT-TYPE
   SYNTAX        StorageType
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Defines the storage type for this object."
   ::= { mplsXCEntry 7 }

mplsXCAdminStatus OBJECT-TYPE
   SYNTAX   INTEGER {
               up(1),     -- ready to pass packets
               down(2),
               testing(3) -- in some test mode
               }
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "The desired operational status of this segment."
```

```
   ::= { mplsXCEntry 8 }

mplsXCOperStatus OBJECT-TYPE
```

```
   SYNTAX   INTEGER {
         up(1),            -- ready to pass packets
         down(2),
         testing(3),       -- in some test mode
         unknown(4),       -- status cannot be determined
                           -- for some reason.
         dormant(5),
         notPresent(6),    -- some component is missing
         lowerLayerDown(7) -- down due to the state of
                           -- lower layer interfaces
      }
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "The actual operational status of this cross-
        connect."
   ::= { mplsXCEntry 9 }

-- End of mplsXCTable


-- Label stack table.
mplsMaxLabelStackDepth  OBJECT-TYPE
   SYNTAX        Integer32 (1..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "The maximum stack depth supported by this LSR."
::= { mplsLsrObjects 10 }

mplsLabelStackIndexNext  OBJECT-TYPE
   SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object contains an appropriate value to be used
        for mplsLabelStackIndex when creating entries in the
        mplsLabelStackTable.  The value 0 indicates that no
        unassigned entries are available.  To obtain an
        mplsLabelStackIndex value for a new entry, the
        manager issues a management protocol retrieval
        operation to obtain the current value of this
        object.  After each retrieval operation, the agent
        should modify the value to reflect the next
        unassigned index.  After a manager retrieves a value
        the agent will determine through its local policy
        when this index value will be made available for
```

```
        reuse."
::= { mplsLsrObjects 11 }
```

```
mplsLabelStackTable OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsLabelStackEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table specifies the label stack to be pushed
        onto a packet, beneath the top label.  Entries into
        this table are referred to from mplsXCTable."
   ::= { mplsLsrObjects 12 }

mplsLabelStackEntry OBJECT-TYPE
   SYNTAX        MplsLabelStackEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table represents one label which is
        to be pushed onto an outgoing packet, beneath the
        top label.  An entry can be created by a network
        administrator or by an SNMP agent as instructed by
        an MPLS signaling protocol."
   INDEX         { mplsLabelStackIndex, mplsLabelStackLabelIndex }
      ::= { mplsLabelStackTable 1 }

MplsLabelStackEntry ::= SEQUENCE {
      mplsLabelStackIndex             Integer32,
      mplsLabelStackLabelIndex        Integer32,
      mplsLabelStackLabel             MplsLabel,
      mplsLabelStackRowStatus         RowStatus,
      mplsLabelStackStorageType       StorageType
   }

mplsLabelStackIndex OBJECT-TYPE
   SYNTAX        Integer32 (1..2147483647)
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "Primary index for this row identifying a stack of
        labels to be pushed on an outgoing packet, beneath
        the top label."
   ::= { mplsLabelStackEntry 1 }

mplsLabelStackLabelIndex OBJECT-TYPE
   SYNTAX        Integer32 (1..2147483647)
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "Secondary index for this row identifying one label
```

of the stack. Note that an entry with a smaller
mplsLabelStackLabelIndex would refer to a label
higher up the label stack and would be popped at a

```
        downstream LSR before a label represented by a
        higher mplsLabelStackLabelIndex at a downstream
        LSR."
   ::= { mplsLabelStackEntry 2 }

mplsLabelStackLabel OBJECT-TYPE
   SYNTAX        MplsLabel
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "The label to pushed."
   ::= { mplsLabelStackEntry 3 }

mplsLabelStackRowStatus OBJECT-TYPE
   SYNTAX        RowStatus
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "For creating, modifying, and deleting this row."
   ::= { mplsLabelStackEntry 4 }

mplsLabelStackStorageType OBJECT-TYPE
   SYNTAX        StorageType
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Defines the storage type for this object."
   ::= { mplsLabelStackEntry 5 }

-- End of mplsLabelStackTable

-- Traffic Parameter table.

mplsTrafficParamIndexNext OBJECT-TYPE
   SYNTAX        Integer32 (0..2147483647)
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
       "This object contains an appropriate value which will
        be used for mplsTrafficParamIndex when creating
        entries in the mplsTrafficParamTable.  The value 0
        indicates that no unassigned entries are available.
        To obtain the mplsTrafficParamIndex value for a new
        entry, the manager issues a management protocol
        retrieval operation to obtain the current value of
        this object.  After each retrieval operation, the
        agent should modify the value to reflect the next
```

unassigned index. After a manager retrieves a value the agent will determine through its local policy when this index value will be made available for

```
        reuse."
   ::= { mplsLsrObjects 13 }

mplsTrafficParamTable OBJECT-TYPE
   SYNTAX        SEQUENCE OF MplsTrafficParamEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "This table specifies the Traffic Parameter objects
        for in and out-segments."
   ::= { mplsLsrObjects 14 }

mplsTrafficParamEntry OBJECT-TYPE
   SYNTAX        MplsTrafficParamEntry
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "An entry in this table represents the TrafficParam
        objects for one or more in or out segments.  A
        single entry can be pointed to by multiple segments
        indicating resource sharing."
   INDEX         { mplsTrafficParamIndex }
      ::= { mplsTrafficParamTable 1 }

MplsTrafficParamEntry ::= SEQUENCE {
      mplsTrafficParamIndex           Integer32,
      mplsTrafficParamMaxRate         MplsBitRate,
      mplsTrafficParamMeanRate        MplsBitRate,
      mplsTrafficParamMaxBurstSize    MplsBurstSize,
      mplsTrafficParamRowStatus       RowStatus,
      mplsTrafficParamStorageType     StorageType
   }

mplsTrafficParamIndex OBJECT-TYPE
   SYNTAX        Integer32 (1..2147483647)
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
       "Uniquely identifies this row of the table.  Note
        that zero represents an invalid index."
   ::= { mplsTrafficParamEntry 1 }

mplsTrafficParamMaxRate OBJECT-TYPE
   SYNTAX        MplsBitRate
   UNITS         "kilobits per second"
   MAX-ACCESS    read-create
   STATUS        current
```

```
    DESCRIPTION
        "Maximum rate in kilobits/second."
    ::= { mplsTrafficParamEntry 2 }
```

```
mplsTrafficParamMeanRate OBJECT-TYPE
   SYNTAX        MplsBitRate
   UNITS         "kilobits per second"
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Mean rate in kilobits/second."
   ::= { mplsTrafficParamEntry 3 }

mplsTrafficParamMaxBurstSize OBJECT-TYPE
   SYNTAX        MplsBurstSize
   UNITS         "bytes"
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "Maximum burst size in bytes."
   ::= { mplsTrafficParamEntry 4 }

mplsTrafficParamRowStatus OBJECT-TYPE
   SYNTAX        RowStatus
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "For creating, modifying, and deleting this row."
   ::= { mplsTrafficParamEntry 5 }

mplsTrafficParamStorageType OBJECT-TYPE
   SYNTAX        StorageType
   MAX-ACCESS    read-create
   STATUS        current
   DESCRIPTION
       "The storage type for this object."
   ::= { mplsTrafficParamEntry 6 }

-- End of mplsTrafficParamTable

-- Notification Configuration

mplsXCTrapEnable OBJECT-TYPE
   SYNTAX        TruthValue
   MAX-ACCESS    read-write
   STATUS        current
   DESCRIPTION
       "If this object is true, then it enables the
        generation of mplsXCUp and mplsXCDown traps,
        otherwise these traps are not emitted."
   DEFVAL { false }
```

```
   ::= { mplsLsrObjects 15 }
```

```
-- Cross-connect.

mplsXCUp NOTIFICATION-TYPE
   OBJECTS     { mplsXCIndex,
                 mplsInSegmentIfIndex,
                 mplsInSegmentLabel,
                 mplsOutSegmentIndex,
                 mplsXCAdminStatus,
                 mplsXCOperStatus }
   STATUS      current
   DESCRIPTION
       "This notification is generated when a
        mplsXCOperStatus object for one of the configured
        cross-connect entries is about to leave the down
        state and transition into some other state (but not
        into the notPresent state).  This other state is
        indicated by the included value of
        mplsXCOperStatus."
   ::= { mplsLsrNotifyPrefix 1 }

mplsXCDown NOTIFICATION-TYPE
   OBJECTS     { mplsXCIndex,
                 mplsInSegmentIfIndex,
                 mplsInSegmentLabel,
                 mplsOutSegmentIndex,
                 mplsXCAdminStatus,
                 mplsXCOperStatus }
   STATUS      current
   DESCRIPTION
       "This notification is generated when a
        mplsXCOperStatus object for one of the configured
        cross-connect entries is about to enter the down
        state from some other state (but not from the
        notPresent state).  This other state is indicated by
        the included value of mplsXCOperStatus."
   ::= { mplsLsrNotifyPrefix 2 }

-- End of notifications.


-- Module compliance.

mplsLsrGroups
   OBJECT IDENTIFIER ::= { mplsLsrConformance 1 }

mplsLsrCompliances
   OBJECT IDENTIFIER ::= { mplsLsrConformance 2 }
```

```
mplsLsrModuleCompliance MODULE-COMPLIANCE
   STATUS current
```

```
   DESCRIPTION
       "Compliance statement for agents that support the
        MPLS LSR MIB."
   MODULE -- this module

      -- The mandatory groups have to be implemented
      -- by all LSRs.  However, they may all be supported
      -- as read-only objects in the case where manual
      -- configuration is unsupported.

      MANDATORY-GROUPS    { mplsInSegmentGroup,
                            mplsOutSegmentGroup,
                            mplsXCGroup,
                            mplsInterfaceGroup,
                            mplsPerfGroup,
                            mplsSegmentDiscontinuityGroup }

      GROUP mplsHCInSegmentPerfGroup
      DESCRIPTION
          "This group is mandatory for those in-segment
           entries for which the object
           mplsInSegmentOutOctets wraps around too
           quickly."

      GROUP mplsHCOutSegmentPerfGroup
      DESCRIPTION
          "This group is mandatory for those out-segment
           entries for which the object
           mplsOutSegmentOctets wraps around too quickly."

      GROUP mplsTrafficParamGroup
      DESCRIPTION
          "This group is mandatory for those LSRs that
           support QoS resource reservation."

      -- Depending on whether the device implements
      -- persistent cross-connects or not one of the
      -- following two groups is mandatory.

      GROUP mplsXCIsPersistentGroup
      DESCRIPTION
          "This group is mandatory for devices which
           support persistent cross-connects.  The
           following constraints apply: mplsXCIsPersistent
           must at least be read-only returning true(2)."

      GROUP mplsXCIsNotPersistentGroup
      DESCRIPTION
```

```
          "This group is mandatory for devices which
           support non-persistent cross-connects.  The
```

```
            following constraints apply: mplsXCIsPersistent
            must at least be read-only returning false(1)."


        -- mplsInterfaceConfTable

        OBJECT      mplsInterfaceConfStorageType
        SYNTAX      INTEGER { other(1) }
        MIN-ACCESS  read-only
        DESCRIPTION
            "Only other(1) needs to be supported."

        -- mplsInSegmentTable

        OBJECT     mplsInSegmentXCIndex
        DESCRIPTION
            "Write access is not required."

        OBJECT      mplsInSegmentNPop
        MIN-ACCESS  read-only
        DESCRIPTION
            "Write access if not required.  This object
             should be set to 1 if it is read-only."

        OBJECT      mplsInSegmentAddrFamily
        DESCRIPTION
            "Write access is not required.  A <value of
             other(0) should be supported because there may
             be cases where the agent may not know about or
             support any address types."

        OBJECT      mplsInSegmentStorageType
        SYNTAX      INTEGER { other(1) }
        MIN-ACCESS  read-only

        DESCRIPTION
            "Only other(1) needs to be supported."

        -- mplsOutSegmentTable

        OBJECT      mplsOutSegmentIfIndex
        MIN-ACCESS  read-only
        DESCRIPTION
            "Write access is not required."

        OBJECT      mplsOutSegmentPushTopLabel
        MIN-ACCESS  read-only
        DESCRIPTION
```

"Write access is not required."

```
      OBJECT      mplsOutSegmentTopLabel
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT      mplsOutSegmentNextHopIpAddrType
      MIN-ACCESS  read-only
      DESCRIPTION
          "ipV6(3) need not be supported."

      OBJECT      mplsOutSegmentNextHopIpv4Addr
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT      mplsOutSegmentNextHopIpv6Addr
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT      mplsOutSegmentRowStatus
      SYNTAX      INTEGER { active(1), notInService(2),
                            createAndGo(4), destroy(6) }
      MIN-ACCESS  read-only
      DESCRIPTION
          "The notReady(3) and createAndWait(5) states need
           not be supported."
      OBJECT      mplsOutSegmentStorageType
      SYNTAX      INTEGER { other(1) }
      MIN-ACCESS  read-only
      DESCRIPTION
          "Only other(1) needs to be supported."

      -- mplsXCTable

      OBJECT      mplsXCLabelStackIndex
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT      mplsXCIsPersistent
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT      mplsXCAdminStatus
      SYNTAX      INTEGER { up(1), down(2) }
      MIN-ACCESS  read-only
```

```
      DESCRIPTION
          "A value of testing(3) need not be supported."
```

```
      OBJECT      mplsXCOperStatus
      SYNTAX      INTEGER { up(1), down(2) }
      DESCRIPTION
          "Only up(1) and down(2) need to be supported."

      OBJECT      mplsXCRowStatus
      SYNTAX      INTEGER { active(1),
                            notInService(2),
                            createAndGo(4),
                            destroy(6) }
      MIN-ACCESS  read-only
      DESCRIPTION
          "The notReady(3) and createAndWait(5) states need
           not be supported."

      OBJECT      mplsXCStorageType
      SYNTAX      INTEGER { other(1) }
      MIN-ACCESS  read-only
      DESCRIPTION
          "Only other(1) needs to be supported."

   ::= { mplsLsrCompliances 1 }


-- Units of conformance.

mplsInterfaceGroup OBJECT-GROUP
   OBJECTS { mplsInterfaceLabelMinIn,
             mplsInterfaceLabelMaxIn,
             mplsInterfaceLabelMinOut,
             mplsInterfaceLabelMaxOut,
             mplsInterfaceTotalBandwidth,
             mplsInterfaceAvailableBandwidth,
             mplsInterfaceLabelParticipationType,
             mplsInterfaceConfStorageType
           }

   STATUS  current
   DESCRIPTION
          "Collection of objects needed for MPLS interface
           configuration and performance information."
   ::= { mplsLsrGroups 1 }


mplsInSegmentGroup  OBJECT-GROUP
             OBJECTS { mplsInSegmentNPop,
             mplsInSegmentAddrFamily,
```

```
            mplsInSegmentXCIndex,
            mplsInSegmentOctets,
```

```
            mplsInSegmentDiscards,
            mplsInSegmentOwner,
            mplsInSegmentRowStatus,
            mplsInSegmentStorageType,
            mplsInSegmentTrafficParamPtr
          }
   STATUS  current
   DESCRIPTION
          "Collection of objects needed to implement an in-
           segment."
   ::= { mplsLsrGroups 2 }

mplsOutSegmentGroup  OBJECT-GROUP
   OBJECTS { mplsOutSegmentIndexNext,
             mplsOutSegmentIfIndex,
             mplsOutSegmentPushTopLabel,
             mplsOutSegmentTopLabel,
             mplsOutSegmentNextHopIpAddrType,
             mplsOutSegmentNextHopIpv4Addr,
             mplsOutSegmentNextHopIpv6Addr,
             mplsOutSegmentXCIndex,
             mplsOutSegmentOwner,
             mplsOutSegmentOctets,
             mplsOutSegmentDiscards,
             mplsOutSegmentErrors,
             mplsOutSegmentRowStatus,
             mplsOutSegmentStorageType,
             mplsOutSegmentTrafficParamPtr
          }
   STATUS  current
   DESCRIPTION
          "Collection of objects needed to implement an out-
           segment."
   ::= { mplsLsrGroups 3 }


mplsXCGroup  OBJECT-GROUP
   OBJECTS { mplsXCIndexNext,
             mplsXCLabelStackIndex,
             mplsXCOwner,
             mplsXCAdminStatus,
             mplsXCOperStatus,
             mplsXCRowStatus,
             mplsXCTrapEnable,
             mplsXCStorageType
          }
   STATUS  current
```

```
   DESCRIPTION
          "Collection of objects needed to implement a
           cross-connect entry."
```

```
   ::= { mplsLsrGroups 4 }

mplsXCOptionalGroup  OBJECT-GROUP
   OBJECTS { mplsXCLspId }
   STATUS  current
   DESCRIPTION
          "Collection of optional objects for implementing
           a cross-connect entry."
   ::= { mplsLsrGroups 5 }

mplsPerfGroup OBJECT-GROUP
   OBJECTS { mplsInSegmentOctets,
             mplsInSegmentPackets,
             mplsInSegmentErrors,
             mplsInSegmentDiscards,
             mplsOutSegmentOctets,
             mplsOutSegmentPackets,
             mplsOutSegmentDiscards,
             mplsInterfaceInLabelsUsed,
             mplsInterfaceFailedLabelLookup,
             mplsInterfaceOutFragments,
             mplsInterfaceOutLabelsUsed
          }

   STATUS  current
   DESCRIPTION
          "Collection of objects providing performance
           information
        about an LSR."
   ::= { mplsLsrGroups 6 }

mplsHCInSegmentPerfGroup OBJECT-GROUP
   OBJECTS { mplsInSegmentHCOctets }
   STATUS  current
   DESCRIPTION
          "Object(s) providing performance information
           specific to out-segments for which the object
           mplsInterfaceInOctets wraps around too quickly."
   ::= { mplsLsrGroups 7 }

mplsHCOutSegmentPerfGroup OBJECT-GROUP
   OBJECTS { mplsOutSegmentHCOctets }
   STATUS  current
   DESCRIPTION
          "Object(s) providing performance information
           specific to out-segments for which the object
           mplsInterfaceOutOctets wraps around too
```

```
            quickly."
    ::= { mplsLsrGroups 8 }
```

```
mplsTrafficParamGroup OBJECT-GROUP
   OBJECTS { mplsTrafficParamIndexNext,
             mplsTrafficParamMaxRate,
             mplsTrafficParamMeanRate,
             mplsTrafficParamMaxBurstSize,
             mplsTrafficParamRowStatus,
             mplsTrafficParamStorageType
           }
   STATUS  current
   DESCRIPTION
          "Object(s) required for supporting QoS resource
           reservation."
   ::= { mplsLsrGroups 9 }

mplsXCIsPersistentGroup OBJECT-GROUP
   OBJECTS { mplsXCIsPersistent }
   STATUS  current
   DESCRIPTION
          "Objects needed to support persistent cross-
           connects."
   ::= { mplsLsrGroups 10 }

mplsXCIsNotPersistentGroup OBJECT-GROUP
   OBJECTS { mplsXCIsPersistent }
   STATUS  current
   DESCRIPTION
          "Objects needed to support non-persistent cross-
           connects."
   ::= { mplsLsrGroups 11 }

mplsLabelStackGroup OBJECT-GROUP
   OBJECTS {mplsLabelStackLabel,
             mplsLabelStackRowStatus,
             mplsLabelStackStorageType,
             mplsMaxLabelStackDepth,
             mplsLabelStackIndexNext }
   STATUS current
   DESCRIPTION
          "Objects needed to support label stacking."
      ::= { mplsLsrGroups 12 }

mplsSegmentDiscontinuityGroup OBJECT-GROUP
   OBJECTS { mplsInSegmentPerfDiscontinuityTime,
             mplsOutSegmentPerfDiscontinuityTime
           }

   STATUS  current
```

```
   DESCRIPTION
       "A collection of objects providing information
        specific to segment discontinuities."
```

```
   ::= { mplsLsrGroups 13 }


mplsLsrNotificationGroup NOTIFICATION-GROUP
   NOTIFICATIONS { mplsInSegmentUp,
                   mplsInSegmentDown,
                   mplsOutSegmentUp,
                   mplsOutSegmentDown,
                   mplsXCUp,
                   mplsXCDown }
   STATUS  current
   DESCRIPTION
          "Set of notifications implemented in this module.
           None is mandatory."
   ::= { mplsLsrGroups 14 }

-- End of MPLS-LSR-MIB
END
```

12. Security Considerations

It is clear that this MIB is potentially useful for monitoring of MPLS LSRs. This MIB can also be used for configuration of certain objects, and anything that can be configured can be incorrectly configured, with potentially disastrous results.

At this writing, no security holes have been identified beyond those that SNMP Security [SNMPArch] is itself intended to address. These relate to primarily controlled access to sensitive information and the ability to configure a device - or which might result from operator error, which is beyond the scope of any security architecture.

There are a number of management objects defined in this MIB which have a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. The use of SNMP Version 3 is recommended over prior versions, for configuration control, as its security model is improved.

SNMPv1 or SNMPv2 are by themselves not a secure environment. Even if the network itself is secure (for example by using IPSec [IPSEC]), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB. It is recommended that the implementers

consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model

[SNMPv3USM] and the View-based Access Control [SNMPv3VACM] is recommended. It is then a customer/user responsibility to ensure that the SNMP entity giving access to an instance of this MIB is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

There are a number of managed objects in this MIB that may contain information that may be sensitive from a business perspective, in that they represent a customer's interface to the MPLS network. Allowing uncontrolled access to these objects could result in malicious and unwanted disruptions of network traffic or incorrect configurations for these customers. There are no objects that are particularly sensitive in their own right, such as passwords or monetary amounts.

13. Acknowledgments

We wish to thank Ron Bonica, Adrian Farrel, Eric Gray, Tim Mancour, Keith McCloghrie, Bala Rajagopalan, Dan Tappan, and Vasanthi Thirumalai.

14. References

[MPLSArch] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", Internet Draft <draft-ietf-mpls-arch-06.txt>, August 1999.

[MPLSFW] Callon, R., Doolan, P., Feldman, N., Fredette, A., Swallow, G., and A. Viswanathan, "A Framework for Multiprotocol Label Switching", Internet Draft <draft-ietf-mpls-framework-05.txt>, September 1999.

[LblStk] Rosen, E., Rekhter, Y., Tappan, D., Farinacci, D., Federokow, G., Li, T., and A. Conta, "MPLS Label Stack Encoding", Internet Draft <draft-ietf-mpls-label-encaps-07.txt>, September 1999.

[Assigned] Reynolds, J., and J. Postel, "Assigned Numbers", RFC 1700, October 1994. See also: http://www.isi.edu/in-notes/iana/assignments/smi-numbers

[IANAFamily] Internet Assigned Numbers Authority (IANA), ADDRESS FAMILY NUMBERS,(http://www.isi.edu/in-

notes/iana/assignements/address-family-numbers),
for MIB see:

ftp://ftp.isi.edu/mib/ianaaddressfamilynumbers.mib

[SNMPArch] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2271, January 1998.

[SMIv1] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, May 1990.

[SNMPv1MIBDef] Rose, M., and K. McCloghrie, "Concise MIB Definitions", RFC 1212, March 1991.

[SNMPv1Traps] M. Rose, "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.

[RFC2572] Case, J., Harrington D., Presuhn R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.

[RFC2574] Blumenthal, U., and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[RFC1905] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[RFC2573] Levi, D., Meyer, P., and B. Stewart, "SNMPv3 Applications", RFC 2573, April 1999.

[RFC2575] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.

[RFC2570] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.

[SMIv2] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[SNMPv2TC] Case, J., McCloghrie, K., Rose, M., and S.

Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, SNMP Research, Inc., Cisco Systems, Inc., January 1996.

[SNMPv2Conf] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.

[SNMPv1] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", RFC 1157, May 1990.

[SNMPv2c] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

[SNMPv2TM] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[SNMPv3MP] Case, J., Harrington D., Presuhn R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2272, January 1998.

[SNMPv3USM] Blumenthal, U., and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[SNMPv2PO] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[SNMPv3App] Levi, D., Meyer, P., and B. Stewart, "SNMPv3 Applications", RFC 2273, January 1998.

[SNMPv3VACM] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.

[IPSEC] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November

1998.

[IFMIB] McCloghrie, K., and F. Kastenholtz, "The Interfaces Group MIB using SMIv2", RFC 2233, Nov. 1997

[ATOMMIB] Tesink, K., "Definitions of Managed Objects for ATM Management", RFC 2515, Feb. 1999

[BCP14] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

15. Authors' Addresses

Cheenu Srinivasan
Tachion Networks, Inc.
Monmouth Park Corporate Center I
Building C, 185 Monmouth Parkway
West Long Branch, NJ 07764
Phone: +1-732-542-7750 x1234
Email: cheenu@tachion.com

Arun Viswanathan
Force10 Networks, Inc.
1440 McCarthy Blvd
Milpitas, CA 95035
Phone: +1-408-571-3516
Email: arun@force10networks.com

Thomas D. Nadeau
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA 01824
Phone: +1-978-244-3051
Email: tnadeau@cisco.com

16. Full Copyright Statement

Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case