

Network Working Group
Request for Comments: 3814
Category: Standards Track

T. Nadeau
Cisco Systems, Inc.
C. Srinivasan
Bloomberg L.P.
A. Viswanathan
Forcel0 Networks, Inc.
June 2004

Multiprotocol Label Switching (MPLS) Forwarding Equivalence
Class To Next Hop Label Forwarding Entry (FEC-To-NHLFE)
Management Information Base (MIB)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2004).

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects for defining, configuring, and monitoring Forwarding Equivalence Class (FEC) to Next Hop Label Forwarding Entry (NHLFE) mappings and corresponding actions for use with Multiprotocol Label Switching (MPLS).

Table of Contents

1.	Introduction	2
2.	Terminology.	3
3.	Conventions Used In This Document.	3
4.	The Internet-Standard Management Framework	3
5.	Outline.	4
5.1.	mplsFTNTable	4
5.1.1.	Advantages of Address Ranges Over CIDR Prefixes.	4
5.2.	mplsFTNMapTable.	5
5.2.1.	Indexing Requirements.	5
5.2.2.	How the Current Indexing Works	5
5.3.	mplsFTNPerfTable	7
6.	Avoiding Retrieval-Modification Interactions	7

7.	Example Illustrating MIB Module Components	8
7.1.	Sample FTN Rules	8
7.2.	Creating FTN Entries and Applying them to Interfaces . . .	9
7.3.	Mapping an FTN Entry to Multiple Interfaces.	10
7.4.	Inserting an Entry Into Existing List.	11
7.5.	Pictorial Tabular Relationship	13
7.6.	Deleting an Entry.	14
8.	The Use of RowPointer.	16
9.	MPLS-FTN-STD-MIB Definitions	16
10.	Security Considerations.	38
11.	IANA Considerations.	39
11.1.	IANA Considerations for MPLS-FTN-STD-MIB	39
12.	References	39
12.1.	Normative References	39
12.2.	Informative References	40
13.	Acknowledgements	41
14.	Authors' Addresses	41
15.	Full Copyright Statement	42

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects for specifying Forwarding Equivalence Class (FEC) to Next Hop Label Forwarding Entry (NHLFE) mappings and corresponding actions for Multiprotocol Label Switching (MPLS).

At the ingress of an MPLS network, packets entering the MPLS domain are assigned to an FEC. Those packets belonging to an FEC are associated with an NHLFE (i.e., MPLS label) via the FEC-to-NHLFE (FTN) mapping [RFC3031]. This relationship defines how ingress LSRs will impose MPLS labels onto incoming packets. It also defines how egress LSRs will decapsulate the MPLS shim header from MPLS packets.

Conceptually, some of the FTN table functionality could be implemented using the Forwarding Information Base (FIB) to map all packets destined for a prefix to an LSP. However, this mapping is coarse in nature.

Similar functionality is already being used in other contexts such as security filters, access filters, and RSVP flow identification. All of these require various combinations of matching based on IP header and upper-layer header information to identify packets for a particular treatment. When packets match a particular rule, a corresponding action is executed on those packets. For example, two popular actions to take when a successful match is identified are allowing the packet to be forwarded or to discard it. However, other

actions are possible, such as modifying the TOS byte, or redirecting a packet to a particular outgoing interface. In the context of MPLS, the possible actions performed by an NHLFE are to redirect packets to either an MPLS Label Switched Path (LSP) or an MPLS Traffic Engineered (TE) Tunnel.

This document attempts to consolidate the various matching requirements and associated action options needed for MPLS into a single specification.

2. Terminology

Although all of the terminology used in this document is either covered in the MPLS Architecture [RFC3031] or in the SNMP Architecture [RFC3411], it is informational to define some immediately pertinent acronyms/terminology here.

MPLS Multiprotocol Label Switching
FEC Forwarding Equivalence Class
NHLFE Next-Hop Label Forwarding Entry
FTN FEC-to-NHLFE
MIB Management Information Base

3. Conventions Used In This Document

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 BCP 14, RFC 2119 [RFC2119].

4. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIV2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

5. Outline

This MIB module resides on any LSR which does the FEC-to-NHLFE mapping in order to map traffic into the MPLS domain. This MIB module consists of three tables:

- `mplsFTNTable` defines the rule base against which incoming packets are matched and defines the actions to be taken on matching packets;
- `mplsFTNMapTable` defines the application of these rules to specific interfaces;
- `mplsFTNPerfTable` provides performance counters for every entry in `mplsFTNTable` that is active on one or more interfaces, on a per-interface basis.

5.1. `mplsFTNTable`

This table allows FEC to NHLFE mappings to be specified. Each entry in this table (also referred to as an "FTN entry" in this document) defines a rule to be applied to incoming packets (on interfaces that the entry is activated on using `mplsFTNMapTable` as explained in Section 5.2) and an action to be taken on matching packets. `mplsFTNTable` allows 6-tuple matching rules based on one or more of source address range, destination address range, source port range, destination port range, IPv4 Protocol field [RFC791] or IPv6 next-header field [RFC2460], and the DiffServ Code Point (DSCP, [RFC2474]) to be specified. Packet redirection is based on an action pointer which points either at an `mplsXCEntry` in MPLS-LSR-STD-MIB [RFC3813] when the NHLFE is a non-TE LSP, or at an `mplsTunnelEntry` in MPLS-TE-STD-MIB [RFC3812] when the NHLFE is the origin of a TE tunnel.

5.1.1. Advantages of Address Ranges Over CIDR Prefixes

One possible way of specifying a set of addresses as part of an FTN rule is to use CIDR prefixes [RFC1519]. We have instead chosen to allow FTN rules to be expressed in terms of address ranges in `mplsFTNTable` because they have the following advantages.

- The number of CIDR prefixes needed to represent some address ranges is very large. For example, we need the following 6 CIDR prefixes to represent the range of addresses [192.0.2.0-192.0.2.62]: 192.0.2.0/27, 192.0.2.32/28, 192.0.2.48/29, 192.0.2.56/30, 192.0.2.60/31, and 192.0.2.62/32. A rule such as "redirect all packets with a source address in the range [192.0.2.0-192.0.2.62] and destination address in the range [192.0.2.128-192.0.2.190] to tunnel #2" would require the creation

of 36 conceptual rows in `mplsFTNTable` if the rules were expressed as CIDR prefixes, but only a single conceptual row would be required if we used address ranges instead.

- Every CIDR prefix can be expressed as a single equivalent address range.
- A particular implementation is free to translate the address ranges specified in `mplsFTNTable` internally to equivalent CIDR prefixes, if it so chooses. However, given that powerful range matching algorithms are available, many implementations may prefer to implement these directly.

5.2. `mplsFTNMapTable`

This table provides the capability to activate or map FTN entries defined in `mplsFTNTable` to specific interfaces in the system. Packets received on an interface are compared against FTN entries in the order in which entries are applied to the interface.

5.2.1. Indexing Requirements

The indexing structure of `mplsFTNMapTable` was designed to satisfy the following requirements.

- We must be able to insert a new entry into an existing list of entries on an interface with a single SET operation. Thus, we must be able to support an insertion operation that does not require manual reindexing of existing entries.
- A management application must be able to traverse entries that have been applied to a particular interface in the order of application. The number of (non-bulk) retrieval operations to obtain this information as dictated by the particular indexing scheme that we choose for `mplsFTNMapTable` must be no more than that dictated by any other indexing scheme. For example, the indexing scheme must not force the Network Management Application to retrieve all the entries in the table and sift through them offline to obtain this information.

5.2.2. How the Current Indexing Works

The natural data-structure for implementing constant time insertions between two existing entries and for supporting in-order traversals is a linked-list.

The chosen indexing structure of `mplsFTNMapTable` makes the entries in the table behave like items in a linked-list. Each conceptual row

has an object, `mplsFTNMapPrevIndex`, which is a pointer to the previous entry that is applied to a particular interface. This object is self-adjusting, i.e., its value is automatically adjusted by the agent, if necessary, after an insertion or deletion operation.

This indexing scheme provides a mechanism to 'insert' an FTN entry between two existing entries already applied on an interface. This is done by specifying the entry after which a new entry should be inserted in `mplsFTNMapPrevIndex`.

Using this linked-list structure, one can retrieve FTN entries in the order of application on a per-interface basis as follows:

- To determine the first FTN entry on an interface with index `ifIndex`, perform a GETNEXT retrieval operation on `mplsFTNMapRowStatus.ifIndex.0.0`; the returned object, if one exists, is (say) `mplsFTNMapRowStatus.ifIndex.0.n` (`mplsFTNMapRowStatus` is the first accessible columnar object in the conceptual row). Then, the index of the first FTN entry applied on this interface is `n`.
- To determine the FTN entry applied to an interface after the one indexed by `n`, perform a GETNEXT retrieval operation on `mplsFTNMapRowStatus.ifIndex.n.0`. If such an entry exists, the returned object would be of the form `mplsFTNMapRowStatus.ifIndex.n.m`. Then, the index of the next FTN entry applied on this interface is `m`.
- If the FTN entry indexed by `n` is the last entry applied to the interface with index `ifIndex`, then the object returned would either be:
 1. `mplsFTNMapRowStatus.ifIndexNext.0.k`, where `ifIndexNext` is the index of the next interface in `ifTable` to which an FTN entry has been applied, in which case `k` is the index of the first FTN entry applied to the interface with index `ifIndexNext`;

or:

2. `mplsFTNMapStorageType.firstIfIndex.0.p`, if there are no more entries in `mplsFTNMapTable`, where `firstIfIndex` is the first entry in `ifTable` to which an FTN entry has been mapped.

The above steps can be used to retrieve all the applied entries on a per-interface basis in application order. Note that the number of retrieval operations is equal to the number of applied FTN entries (i.e., the minimum number of GETNEXT operations needed using any indexing scheme).

Also note that we could not have created this linked-list structure using a 'next' pointer object instead of the 'previous' pointer object that we chose because this would not allow us to determine the first FTN entry that has been mapped to a specific interface using a single SNMP (non-bulk) retrieval operation.

The use of this indexing structure is further illustrated using an example in Section 7.

5.3. mplsFTNPerfTable

If an FTN entry has been applied to one or more interfaces, this table provides high-capacity performance counters to monitor each such FTN entry on a per-interface basis.

6. Avoiding Retrieval-Modification Interactions

The problem of an ongoing traversal or retrieval operation on an SNMP table being affected by a concurrent modification operation on that table is not unique to this MIB module. However, it is useful to note that a cautious application can keep track of the state of the modifiable tables in this MIB module using the objects `mplsFTNTableLastChanged` and `mplsFTNMapTableLastChanged`.

For instance, before performing a traversal of `mplsFTNMapTable`, the application should retrieve the value of `mplsFTNMapTableLastChanged`. Each subsequent GETNEXT operation on the table should include this object as well. For example, `GETNEXT(mplsFTNMapTableLastChanged.0, mplsFTNMapRowStatus.ifIndex.n.0)` can be used to:

- Determine the FTN entry after the one indexed by `n` (in linked-list order) mapped to the interface with index `ifIndex`, as explained in Section 5.2.2;
- Verify that the value of `mplsFTNMapTable` has not been modified during the retrieval process by comparing the value of `mplsFTNMapTableLastChanged` retrieved by this operation with the value retrieved before the traversal was begun.

Using this technique, an application can ensure the validity of the retrieved information with minimal overhead. This is particularly important while retrieving information from frequently modified tables.

7. Example Illustrating MIB Module Components

In this section, we use an example to illustrate how the objects defined in MPLS-FTN-STD-MIB work together to perform FEC to NHLFE mapping.

Note that for the various table entries involved in this example, we only show the objects that help illustrate each case.

7.1. Sample FTN Rules

Suppose that we wish to activate the following two FTN rules.

Rule #1: On interface `ifIndex = 1`, redirect packets with source IPv4 address matching `192.0.2.63` to an LSP with outgoing `ifIndex = 50` and outgoing label = `150` where the specified LSP is represented by the following entries in `mplsXCTable` and `mplsOutSegmentTable`.

In `mplsXCTable`:

```
{
  mplsXCIndex = 0x02,
  mplsXCInSegmentIndex = 0x00,
  mplsXCOutSegmentIndex = 0x03,
  mplsXCLabelStackIndex = 0
}
```

The value `0x00` for `mplsXCInSegmentIndex` represents an originating LSP [RFC3813].

In `mplsOutSegmentTable`:

```
{
  mplsOutSegmentIndex = 0x03,
  mplsOutSegmentIfIndex = 50,
  mplsOutSegmentPushTopLabel = true,
  mplsOutSegmentTopLabel = 150
}
```

Rule #2: On interface `ifIndex = 1`, redirect packets with destination IPv4 addresses in the range [`192.0.2.32`, `192.0.2.96`] to tunnel #4, where the specified tunnel is represented by the following entry in `mplsTunnelTable`:


```

{
  mplsTunnelIndex = 4,
  -- primary tunnel
  mplsTunnelInstance = 0,
  mplsTunnelIngressLSRID = 192.0.2.1,
  mplsTunnelEgressLSRID = 192.0.2.2
}

```

7.2. Creating FTN Entries and Applying them to Interfaces

The action "redirect packets with source IPv4 address matching 192.0.2.63 to an LSP with outgoing ifIndex = 50 and outgoing label = 150" in Rule #1 can be implemented by the following entry in mplsFTNTable:

```

{
  mplsFTNIndex = 1,
  mplsFTNDescr = "Rule #1",
  -- source address only
  mplsFTNMask = 0x80,
  mplsFTNAddrType = ipv4,
  mplsFTNSourceAddrMin = 192.0.2.63,
  mplsFTNSourceAddrMax = 192.0.2.63,
  mplsFTNActionType = redirectLsp(1),
  mplsFTNActionPointer = mplsXCLspId.1.2.1.0.1.3
}

```

This indicates to which LSP the LSR should redirect packets by setting mplsFTNActionPointer to the first accessible columnar object instance in mplsXCEntry that corresponds of the LSP to use, in this case mplsXCLspId.1.2.1.0.1.3.

This action is then activated on "interface ifIndex = 1" by the following entry in mplsFTNMapTable to complete the implementation of Rule #1:

```

{
  -- apply rule to interface ifIndex = 1
  mplsFTNMapIndex = 1,
  -- first FTN entry on this interface
  mplsFTNPrevIndex = 0,
  -- index of current entry in mplsFTNTable, i.e., Rule #1
  mplsFTNMapCurrIndex = 1
}

```

The action "redirect packets with destination IPv4 addresses in the range [192.0.2.32, 192.0.2.96] to tunnel #4" in Rule #2 can be implemented by the following entry in mplsFTNTable:

```

{
  mplsFTNIndex = 2,
  mplsFTNDescr = "Rule #2",
  -- destination address only
  mplsFTNMask = 0x40,
  mplsFTNAddrType = ipv4,
  mplsFTNDestAddrMin = 192.0.2.32,
  mplsFTNDestAddrMax = 192.0.2.96,
  mplsFTNActionType = redirectTunnel(2),
  mplsFTNActionPointer = mplsTunnelName.4.0.3221225985.3221225986
}

```

where 3221225985 and 3221225986 are representations of the addresses 192.0.2.1 and 192.0.2.2, respectively, as Unsigned32 (the underlying data type) entities.

This rule needs to be activated on "interface ifIndex = 1" after Rule #1 which was previously activated on this interface. This is done by the following entry in mplsFTNMapTable to complete the implementation of Rule #2:

```

{
  -- apply rule to interface ifIndex = 1
  mplsFTNMapIndex = 1,
  -- insert after Rule #1 (mplsFTNIndex = 1)
  mplsFTNPrevIndex = 1,
  -- index of current entry in mplsFTNTable, i.e., Rule #2
  mplsFTNMapCurrIndex = 2
}

```

7.3. Mapping an FTN Entry to Multiple Interfaces

Suppose we now wish to activate the following rule:

```

Rule #2b: On interface ifIndex = 2, redirect packets with
destination IPv4 addresses in the range [192.0.2.32, 192.0.2.96]
to tunnel #4.

```

Notice that the FEC and corresponding action associated with this rule (i.e., "redirect packets with destination IPv4 addresses in the range [192.0.2.32, 192.0.2.96] to tunnel #4") are the same as that associated with Rule #2. Hence, we can reuse the existing entry with mplsFTNIndex = 2 from mplsFTNTable.

However, we have to create the following new entry in mplsFTNMapTable to activate this FTN entry as the first one on the interface with ifIndex = 2.

```
{
  -- apply rule to interface ifIndex = 2
  mplsFTNMapIndex = 2,
  -- first FTN entry on this interface
  mplsFTNPrevIndex = 0,
  -- index of current entry in mplsFTNTable
  mplsFTNMapCurrIndex = 2
}
```

7.4. Inserting an Entry Into Existing List

At a later point, suppose that we wish to introduce the following Rule between Rules #1 and #2.

Rule #3: On interface ifIndex = 1, redirect all packets with destination IPv4 address matching the prefix 192.0.2.32/28 to tunnel #3, where the tunnel we wish to redirect traffic to is represented by the following entry in mplsTunnelTable:

```
{
  mplsTunnelIndex = 3,
  -- primary tunnel
  mplsTunnelInstance = 0,
  mplsTunnelIngressLSRID = 192.0.2.3,
  mplsTunnelEgressLSRID = 192.0.2.4
}
```

Note that the ordering of the rules on a particular interface is critical since the range of addresses specified in Rule #3 is a subset of the ones specified in Rule #2.

Without the linked-list style insertion feature supported by mplsFTNMapTable, we would possibly have had to reindex existing entries (or plan for such changes by leaving sufficient gaps between indexes, something that only postpones the problem). With the existing tables, we solve this problem by creating the following entries.

We implement the phrase "redirect all packets with destination IPv4 address matching the prefix 1.4.0.0/16 to tunnel #3" in Rule #3 by creating the following entry in mplsFTNTable:

```
{
  mplsFTNIndex = 3,
  mplsFTNDescr = "Rule #3",
  -- destination address only
  mplsFTNMask = 0x40,
  mplsFTNAddrType = ipv4,
  -- address range equivalent to CIDR prefix 192.0.2.32/28
  mplsFTNDestAddrMin = 192.0.2.32,
  mplsFTNDestAddrMax = 192.0.2.47,
  mplsFTNActionType = redirectTunnel,
  mplsFTNActionPointer = mplsTunnelName.3.0.3221225987.3221225988
}
```

where 3221225987 and 3221225988 are representations of the addresses 192.0.2.3 and 192.0.2.4, respectively, as Unsigned32 (the underlying data type) entities.

We next insert this rule in mplsFTNMapTable just after Rule #1 as follows:

```
{
  -- apply rule to interface ifIndex = 1
  mplsFTNMapIndex = 1,
  -- insert after Rule #1 (mplsFTNIndex = 1)
  mplsFTNPrevIndex = 1,
  -- index of current entry in mplsFTNTable i.e., Rule #3
  mplsFTNMapCurrIndex = 3
}
```

After the insertion of Rule #3 in mplsFTNMapTable, the 'previous' pointer object mplsFTNMapPrevIndex of the next entry (corresponding to Rule #2) adjusts automatically to point to this entry.

Note that, of the existing entries in the table, the only one that is impacted by an insertion operation is the entry on that particular interface immediately after the newly inserted one, if one exists. None of the other entries in mplsFTNMapTable are impacted. For instance, in this particular example, when the entry for Rule #3 was inserted between those for Rules #1 and #2, the entries for Rules #1 and #2b were not impacted.

7.5. Pictorial Tabular Relationship

At this point, the relationship between different table entries can be represented pictorially as follows. For each conceptual row instance, we show the table that it belongs to, along with its indices in parentheses. (Note that various conceptual rows are depicted in a way that is convenient for showing the interrelationships and are not necessarily in lexicographical order.)

```

    ifTable, The Interfaces Group MIB [RFC2863]:
+--> ifEntry (1)
    |   (ifIndex = 1)
    |
    |   mplsFTNMapTable:
    |   mplsFTNMapEntry (1.0.1): <-----+
+<-- (mplsFTNMapIndex = 1,
    |   mplsFTNMapPrevIndex = 0, ---> (NULL)
    |   mplsFTNMapCurrIndex = 1) -----+
    |
    |   mplsFTNMapEntry (1.1.3): <-----+
+<-- (mplsFTNMapIndex = 1,
    |   mplsFTNMapPrevIndex = 1, ----->+
    |   mplsFTNMapCurrIndex = 3) -----+
    |
    |   mplsFTNMapEntry (1.3.2): <-----+
+<-- (mplsFTNMapIndex = 1,
    |   mplsFTNMapPrevIndex = 3, ----->+
    |   mplsFTNMapCurrIndex = 2) -----+
    |
    |   mplsFTNTable:
    |   mplsFTNEntry (2):
+<--> (mplsFTNIndex = 2) <-----+
    |
    |   mplsFTNEntry (3):
    |   (mplsFTNIndex = 3) <-----+
    |
    |   mplsFTNEntry (1):
    |   (mplsFTNIndex = 1) <-----+
    |
    |   mplsFTNPerfTable:
    |   mplsFTNPerfEntry (1.2):
    |   (mplsFTNPerfIndex = 1,
    |   mplsFTNPerfCurrIndex = 2) -----+
    |
    |   mplsFTNPerfEntry (1.3):
    |   (mplsFTNPerfIndex = 1,
    |   mplsFTNPerfCurrIndex = 3) -----+

```

```

| mplsFTNPerfEntry (1.1): |
| (mplsFTNPerfIndex = 1, |
| mplsFTNPerfCurrIndex = 1) -----+
|
| mplsFTNPerfEntry (2.2): |
| (mplsFTNPerfIndex = 2, |
| mplsFTNPerfCurrIndex = 2) -----+
|
| ifTable, The Interfaces Group MIB [RFC2863]: |
+----> ifEntry (2): |
| (ifIndex = 2) |
|
| mplsFTNMapEntry (2.1.2): <-----+
+----- (mplsFTNMapIndex = 2 |
| mplsFTNMapPrevIndex = 0 ----> (NULL) |
+----- mplsFTNMapCurrIndex = 2)

```

7.6. Deleting an Entry

Let us next look at how we can remove the recently applied Rule #3 and how the existing conceptual rows behave in this situation.

The conceptual row corresponding to the application of Rule #3 to interface ifIndex = 1 has the following index values: mplsFTNMapIndex = 1, mplsFTNMapPrevIndex = 1, and mplsFTNMapCurrIndex = 3. To delete this conceptual row, the Network Management Application performs a SET operation setting the object instance mplsFTNMapRowStatus.1.1.3 to the value destroy(6). The agent then destroys this conceptual row. It also automatically adjusts the object instance of mplsFTNMapPrevIndex corresponding to Rule #2 from the value 3 (i.e., pointing to the recently destroyed Rule #3) to the value 1 (i.e., to Rule #1).

At this point, the rules applied to interface ifIndex = 1 are Rule #1 and Rule #2, in that order. The relationship between different table entries can be represented pictorially as follows.

```

    ifTable, The Interfaces Group MIB [RFC2863]:
+--> ifEntry (1)
    |   (ifIndex = 1)
    |
    |   mplsFTNMapTable:
    |   mplsFTNMapEntry (1.0.1): <-----+
+<-- (mplsFTNMapIndex = 1,
    |   mplsFTNMapPrevIndex = 0, ----> (NULL)
    |   mplsFTNMapCurrIndex = 1) -----+
    |
    |   mplsFTNMapEntry (1.1.2): <-----+
+<-- (mplsFTNMapIndex = 1,
    |   mplsFTNMapPrevIndex = 1, ---->
    |   mplsFTNMapCurrIndex = 2) -----+
    |
    |   mplsFTNTable:
    |   mplsFTNEntry (2):
+----> (mplsFTNIndex = 2) <-----+
    |
    |   mplsFTNEntry (3):
    |   (mplsFTNIndex = 3)
    |
    |   mplsFTNEntry (1):
    |   (mplsFTNIndex = 1) <-----+
    |
    |   mplsFTNPerfTable:
    |   mplsFTNPerfEntry (1.2):
    |   (mplsFTNPerfIndex = 1,
    |   mplsFTNPerfCurrIndex = 2) -----+
    |
    |   mplsFTNPerfEntry (1.1):
    |   (mplsFTNPerfIndex = 1,
    |   mplsFTNPerfCurrIndex = 1) -----+
    |
    |   mplsFTNPerfEntry (2.2):
    |   (mplsFTNPerfIndex = 2,
    |   mplsFTNPerfCurrIndex = 2) -----+
    |
    |   ifTable, The Interfaces Group MIB [RFC2863]:
+----> ifEntry (2):
    |   (ifIndex = 2)
    |
    |   mplsFTNMapEntry (2.1.2): <-----+
+----- (mplsFTNMapIndex = 2
    |   mplsFTNMapPrevIndex = 0 ----> (NULL)
+----- mplsFTNMapCurrIndex = 2)

```

Note that the FTN entry for Rule #3 still exists in `mplsFTNTable` at this point but is not referenced by any conceptual row in `mplsFTNMapTable` or `mplsFTNPerfTable`.

Also note that the deletion of an entry from `mplsFTNMapTable` only impacts the entry on that particular interface immediately after the deleted entry, if one exists. None of the other conceptual rows in `mplsFTNMapTable` are impacted. For instance, in this particular example, when the entry for Rule #3 was deleted, the entries for Rules #1 and #2b were not impacted.

8. The Use of RowPointer

`RowPointer` is a textual convention used to identify a conceptual row in a conceptual table in a MIB by pointing to the first accessible object. In this MIB module, in `mplsFTNTable`, the `RowPointer` object `mplsFTNActionPointer` indicates the LSP or TE Tunnel to redirect packets matching an FTN entry to. This object MUST point to the first instance of the first accessible columnar object in the appropriate conceptual row in order to allow the manager to find the appropriate corresponding entry in either `MPLS-LSR-STD-MIB` [RFC3813] or `MPLS-TE-STD-MIB` [RFC3812]. If this object returns `zeroDotZero`, it implies that there is no currently defined action that is associated with that particular FTN entry.

9. MPLS-FTN-STD-MIB Definitions

```
MPLS-FTN-STD-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```

MODULE-IDENTITY, OBJECT-TYPE, Unsigned32, Counter64, Integer32
    FROM SNMPv2-SMI -- [RFC2578]
RowStatus, StorageType, RowPointer,
TEXTUAL-CONVENTION, TimeStamp
    FROM SNMPv2-TC -- [RFC2579]
MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-CONF -- [RFC2580]
InterfaceIndexOrZero,
ifGeneralInformationGroup, ifCounterDiscontinuityGroup
    FROM IF-MIB -- [RFC2863]
SnmpAdminString
    FROM SNMP-FRAMEWORK-MIB -- [RFC3411]
Dscp
    FROM DIFFSERV-DSCP-TC -- [RFC3289]
InetAddressType, InetAddress, InetPortNumber
    FROM INET-ADDRESS-MIB -- [RFC3291]
mplsStdMIB
    FROM MPLS-TC-STD-MIB -- [RFC3811]

```


;

mplsFTNStdMIB MODULE-IDENTITY

LAST-UPDATED "200406030000Z" -- June 6, 2004

ORGANIZATION "Multiprotocol Label Switching (MPLS) Working Group"

CONTACT-INFO

"

Thomas D. Nadeau

Postal: Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA 01824

Tel: +1-978-244-3051

Email: tnadeau@cisco.com

Cheenu Srinivasan

Postal: Bloomberg L.P.
499 Park Avenue
New York, NY 10022

Tel: +1-212-893-3682

Email: cheenu@bloomberg.net

Arun Viswanathan

Postal: Forcel0 Networks, Inc.
1440 McCarthy Blvd
Milpitas, CA 95035

Tel: +1-408-571-3516

Email: arunv@forcel0networks.com

IETF MPLS Working Group email: mpls@uu.net"

DESCRIPTION

"Copyright (C) The Internet Society (2004). The initial version of this MIB module was published in RFC 3814. For full legal notices see the RFC itself or see:

<http://www.ietf.org/copyrights/ianamib.html>

This MIB module contains managed object definitions for specifying FEC to NHLFE (FTN) mappings and corresponding performance for MPLS."

-- Revision history.

REVISION

"200406030000Z" -- June 3, 2004

DESCRIPTION

"Initial version issued as part of RFC 3814."

```

 ::= { mplsStdMIB 8 }

-- TEXTUAL-CONVENTIONS used in this MIB.
MplsFTNEntryIndex ::= TEXTUAL-CONVENTION
  STATUS          current
  DESCRIPTION
    "Index for an entry in mplsFTNTable."
  SYNTAX          Unsigned32 (1..4294967295)

MplsFTNEntryIndexOrZero ::= TEXTUAL-CONVENTION
  STATUS          current
  DESCRIPTION
    "Index for an entry in mplsFTNTable or the special value
    zero. The value zero is object-specific and must
    therefore be defined as part of the description of any
    object which uses this syntax. Examples of the usage
    of zero might include situations when none or all
    entries in mplsFTNTable need to be referenced."
  SYNTAX          Unsigned32 (0..4294967295)

-- Top-Level Components of this MIB.

mplsFTNNotifications OBJECT IDENTIFIER ::= { mplsFTNStdMIB 0 }
mplsFTNObjects        OBJECT IDENTIFIER ::= { mplsFTNStdMIB 1 }
mplsFTNConformance   OBJECT IDENTIFIER ::= { mplsFTNStdMIB 2 }

-- Next free index in mplsFTNTable.
mplsFTNIndexNext OBJECT-TYPE
  SYNTAX          MplsFTNEntryIndexOrZero
  MAX-ACCESS      read-only
  STATUS          current
  DESCRIPTION
    "This object contains the next available valid value to
    be used for mplsFTNIndex when creating entries in the
    mplsFTNTable.

    When creating a new conceptual row (configuration
    entry) in mplsFTNTable with an SNMP SET operation the
    command generator (Network Management Application) must
    first issue a management protocol retrieval operation
    to obtain the current value of this object.

    If the command responder (agent) does not wish to allow
    creation of more entries in mplsFTNTable, possibly
    because of resource exhaustion, this object MUST return
    a value of 0.

    If a non-zero value is returned the Network Management

```

Application must determine whether the value is indeed still unused since two Network Management Applications may attempt to create a row simultaneously and use the same value.

If it is currently unused and the SET succeeds, the agent MUST change the value of this object to a currently unused non-zero value (according to an implementation specific algorithm) or zero (if no further row creation will be permitted).

If the value is in use, however, the SET fails and the Network Management Application must then reread this object to obtain a new usable value."

```
::= { mplsFTNObjects 1 }
```

```
-- Last time an object in mplsFTNTable changed.
```

```
mplsFTNTableLastChanged OBJECT-TYPE
```

```
SYNTAX          TimeStamp
```

```
MAX-ACCESS      read-only
```

```
STATUS          current
```

```
DESCRIPTION
```

"Indicates the last time an entry was added, deleted or modified in mplsFTNTable. Management stations should consult this object to determine if mplsFTNTable requires their attention. This object is particularly useful for applications performing a retrieval on mplsFTNTable to ensure that the table is not modified during the retrieval operation."

```
::= { mplsFTNObjects 2 }
```

```
-- Table of FTN entries.
```

```
mplsFTNTable OBJECT-TYPE
```

```
SYNTAX          SEQUENCE OF MplsFTNEntry
```

```
MAX-ACCESS      not-accessible
```

```
STATUS          current
```

```
DESCRIPTION
```

"This table contains the currently defined FTN entries. This table allows FEC to NHLFE mappings to be specified. Each entry in this table defines a rule to be applied to incoming packets (on interfaces that the FTN entry is activated on using mplsFTNMapTable) and an action to be taken on matching packets (mplsFTNActionPointer).

This table supports 6-tuple matching rules based on one or more of source address range, destination address range, source port range, destination port range, IPv4

Protocol field or IPv6 next-header field and the DiffServ Code Point (DSCP) to be specified.

The action pointer points either to instance of `mplsXCEnt` in MPLS-LSR-STD-MIB when the NHLFE is a non-TE LSP, or to an instance of `mplsTunnelEntry` in the MPLS-TE-STD-MIB when the NHLFE is an originating TE tunnel."

REFERENCE

"J. Postel, Internet Protocol, RFC 791, STD 5, September 1981

Deering, S., and R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, RFC 2460, December 1998

Nichols, K, Blake, S., Baker, F. and D. Black, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, December 1998

Srinivasan, C., A. Viswanathan, and T. Nadeau, MPLS Label Switch Router Management Information Base, RFC 3813

Srinivasan, C., A. Viswanathan, and T. Nadeau, MPLS Traffic Engineering Management Information Base, RFC 3812"

```
::= { mplsFTNObjects 3 }
```

```
mplsFTNEntry OBJECT-TYPE
```

```
SYNTAX          MplsFTNEntry
MAX-ACCESS      not-accessible
STATUS          current
```

```
DESCRIPTION
```

```
"Each entry represents one FTN entry which defines a
rule to compare incoming packets with and an action to
be taken on matching packets."
```

```
INDEX { mplsFTNIndex }
```

```
::= { mplsFTNTable 1 }
```

```
MplsFTNEntry ::= SEQUENCE {
```

```
    mplsFTNIndex          MplsFTNEntryIndex,
    mplsFTNRowStatus      RowStatus,
    mplsFTNDescr          SnmpAdminString,
    mplsFTNMask           BITS,
    mplsFTNAddrType       InetAddressType,
    mplsFTNSourceAddrMin  InetAddress,
    mplsFTNSourceAddrMax  InetAddress,
```

```

    mplsFTNDestAddrMin      InetAddress,
    mplsFTNDestAddrMax      InetAddress,
    mplsFTNSourcePortMin    InetPortNumber,
    mplsFTNSourcePortMax    InetPortNumber,
    mplsFTNDestPortMin      InetPortNumber,
    mplsFTNDestPortMax      InetPortNumber,
    mplsFTNProtocol          Integer32,
    mplsFTNDscp              Dscp,
    mplsFTNActionType        INTEGER,
    mplsFTNActionPointer     RowPointer,
    mplsFTNStorageType       StorageType
}

mplsFTNIndex OBJECT-TYPE
    SYNTAX          MplsFTNEntryIndex
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This is the unique index for a conceptual row in
        mplsFTNTable.

        To create a new conceptual row in mplsFTNTable a
        Network Management Application SHOULD retrieve the
        current value of mplsFTNIndexNext to determine the next
        valid available value of mplsFTNIndex."
    ::= { mplsFTNEntry 1 }

mplsFTNRowStatus OBJECT-TYPE
    SYNTAX          RowStatus
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Used for controlling the creation and deletion of this
        row. All writeable objects in this row may be modified
        at any time. If a Network Management Application
        attempts to delete a conceptual row by setting this
        object to 'destroy' and there are one or more entries
        in mplsFTNMapTable pointing to the row (i.e., when
        mplsFTNIndex of the conceptual row being deleted is
        equal to mplsFTNMapCurrIndex for one or more entries in
        mplsFTNMapTable), the agent MUST also destroy the
        corresponding entries in mplsFTNMapTable."
    ::= { mplsFTNEntry 2 }

mplsFTNDescr OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-create
    STATUS          current

```

DESCRIPTION

"The description of this FTN entry. Since the index for this table has no particular significance or meaning, this object should contain some meaningful text that an operator could use to further distinguish entries in this table."

```
::= { mplsFTNEntry 3 }
```

mplsFTNMask OBJECT-TYPE

```
SYNTAX          BITS {
                  sourceAddr(0),
                  destAddr(1),
                  sourcePort(2),
                  destPort(3),
                  protocol(4),
                  dscp(5)
                }
```

```
MAX-ACCESS      read-create
```

```
STATUS          current
```

DESCRIPTION

"This bit map indicates which of the fields described next, namely source address range, destination address range, source port range, destination port range, IPv4 Protocol field or IPv6 next-header field and Differentiated Services Code Point (DSCP) is active for this FTN entry. If a particular bit is set to zero then the corresponding field in the packet MUST be ignored for comparison purposes."

```
::= { mplsFTNEntry 4 }
```

mplsFTNAddrType OBJECT-TYPE

```
SYNTAX          InetAddressType
```

```
MAX-ACCESS      read-create
```

```
STATUS          current
```

DESCRIPTION

"This object determines the type of address contained in the source and destination address objects (mplsFTNSourceAddrMin, mplsFTNSourceAddrMax, mplsFTNDestAddrMin and mplsFTNDestAddrMax) of a conceptual row.

This object MUST NOT be set to unknown(0) when mplsFTNMask has bit positions sourceAddr(0) or destAddr(1) set to one.

When both these bit positions of mplsFTNMask are set to zero the value of mplsFTNAddrType SHOULD be set to unknown(0) and the corresponding source and destination

```
        address objects SHOULD be set to zero-length strings."
 ::= { mplsFTNEntry 5 }

mplsFTNSourceAddrMin OBJECT-TYPE
    SYNTAX          InetAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The lower end of the source address range. The type of
         this object is determined by the corresponding
         mplsFTNAddrType object."
 ::= { mplsFTNEntry 6 }

mplsFTNSourceAddrMax OBJECT-TYPE
    SYNTAX          InetAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The upper end of the source address range. The type of
         this object is determined by the corresponding
         mplsFTNAddrType object."
 ::= { mplsFTNEntry 7 }

mplsFTNDestAddrMin OBJECT-TYPE
    SYNTAX          InetAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The lower end of the destination address range. The
         type of this object is determined by the corresponding
         mplsFTNAddrType object."
 ::= { mplsFTNEntry 8 }

mplsFTNDestAddrMax OBJECT-TYPE
    SYNTAX          InetAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The higher end of the destination address range. The
         type of this object is determined by the corresponding
         mplsFTNAddrType object."
 ::= { mplsFTNEntry 9 }

mplsFTNSourcePortMin OBJECT-TYPE
    SYNTAX          InetPortNumber
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
```

```
        "The lower end of the source port range."
DEFVAL { 0 }
 ::= { mplsFTNEntry 10 }

mplsFTNSourcePortMax OBJECT-TYPE
SYNTAX      InetPortNumber
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The higher end of the source port range "
DEFVAL { 65535 }
 ::= { mplsFTNEntry 11 }

mplsFTNDestPortMin OBJECT-TYPE
SYNTAX      InetPortNumber
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The lower end of the destination port range."
DEFVAL { 0 }
 ::= { mplsFTNEntry 12 }

mplsFTNDestPortMax OBJECT-TYPE
SYNTAX      InetPortNumber
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The higher end of the destination port range."
DEFVAL { 65535 }
 ::= { mplsFTNEntry 13 }

mplsFTNProtocol OBJECT-TYPE
SYNTAX      Integer32 (0..255)
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The IP protocol to match against the IPv4 protocol
    number or IPv6 Next-Header number in the packet. A
    value of 255 means match all. Note that the protocol
    number of 255 is reserved by IANA, and Next-Header
    number of 0 is used in IPv6."
DEFVAL { 255 }
 ::= { mplsFTNEntry 14 }

mplsFTNDscp OBJECT-TYPE
SYNTAX      Dscp
MAX-ACCESS  read-create
STATUS      current
```


DESCRIPTION

"The contents of the DSCP field."

REFERENCE

"Nichols, K., Blake, S., Baker, F. and D. Black,
Definition of the Differentiated Services Field (DS
Field) in the IPv4 and IPv6 Headers, RFC 2474, December
1998."

::= { mplsFTNEntry 15 }

mplsFTNActionType OBJECT-TYPE

```
SYNTAX      INTEGER {
                redirectLsp(1),    -- redirect into LSP
                redirectTunnel(2) -- redirect into tunnel
            }
```

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The type of action to be taken on packets matching this
FTN entry."

::= { mplsFTNEntry 16 }

mplsFTNActionPointer OBJECT-TYPE

```
SYNTAX      RowPointer
```

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"If mplsFTNActionType is redirectLsp(1), then this
object MUST contain zeroDotZero or point to a instance
of mplsXCEntry indicating the LSP to redirect matching
packets to.

If mplsFTNActionType is redirectTunnel(2), then this
object MUST contain zeroDotZero or point to a instance
of mplsTunnelEntry indicating the MPLS TE tunnel to
redirect matching packets to.

If this object points to a conceptual row instance in a
table consistent with mplsFTNActionType but this
instance does not currently exist then no action will
be taken on packets matching such an FTN entry till
this instance comes into existence.

If this object contains zeroDotZero then no action will
be taken on packets matching such an FTN entry till it
is populated with a valid pointer consistent with the
value of mplsFTNActionType as explained above."

::= { mplsFTNEntry 17 }

```

mplsFTNStorageType OBJECT-TYPE
    SYNTAX          StorageType
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "The storage type for this FTN entry. Conceptual rows
        having the value 'permanent' need not allow write-
        access to any columnar objects in the row."
    DEFVAL { nonVolatile }
    ::= { mplsFTNEntry 18 }

-- End of mplsFTNTable.

-- Last time an object in mplsFTNMapTable changed.

mplsFTNMapTableLastChanged OBJECT-TYPE
    SYNTAX          TimeStamp
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Indicates the last time an entry was added, deleted or
        modified in mplsFTNMapTable. Management stations should
        consult this object to determine if the table requires
        their attention. This object is particularly useful
        for applications performing a retrieval on
        mplsFTNMapTable to ensure that the table is not
        modified during the retrieval operation."
    ::= { mplsFTNObjects 4 }

-- FTN to interface mapping table.

mplsFTNMapTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF MplsFTNMapEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This table contains objects which provide the
        capability to apply or map FTN rules as defined by
        entries in mplsFTNTable to specific interfaces in the
        system. FTN rules are compared with incoming packets
        in the order in which they are applied on an interface.

        The indexing structure of mplsFTNMapTable is as
        follows.

        - mplsFTNMapIndex indicates the interface to which the
        rule is being applied. A value of 0 represents the
        application of the rule to all interfaces."

```

- mplsFTNMapPrevIndex specifies the rule on the interface prior to the one being applied. A value of 0 specifies that the rule is being inserted at the head of the list of rules currently applied to the interface.
- mplsFTNMapCurrIndex is the index in mplsFTNTable corresponding to the rule being applied.

This indexing structure makes the entries in the table behave like items in a linked-list. The object mplsFTNMapPrevIndex in each conceptual row is a pointer to the previous entry that is applied to a particular interface. This allows a new entry to be 'inserted' at an arbitrary position in a list of entries currently applied to an interface. This object is self-adjusting, i.e., its value is automatically adjusted by the agent, if necessary, after an insertion or deletion operation.

Using this linked-list structure, one can retrieve FTN entries in the order of application on a per-interface basis as follows:

- To determine the first FTN entry on an interface with index ifIndex perform a GETNEXT retrieval operation on mplsFTNMapRowStatus.ifIndex.0.0; the returned object, if one exists, is (say) mplsFTNMapRowStatus.ifIndex.0.n (mplsFTNMapRowStatus is the first accessible columnar object in the conceptual row). Then the index of the first FTN entry applied on this interface is n.
- To determine the FTN entry applied to an interface after the one indexed by n perform a GETNEXT retrieval operation on mplsFTNMapRowStatus.ifIndex.n.0. If such an entry exists the returned object would be of the form mplsFTNMapRowStatus.ifIndex.n.m. Then the index of the next FTN entry applied on this interface is m.
- If the FTN entry indexed by n is the last entry applied to the interface with index ifIndex then the object returned would either be:
 1. mplsFTNMapRowStatus.ifIndexNext.0.k, where ifIndexNext is the index of the next interface in

ifTable to which an FTN entry has been applied, in which case k is the index of the first FTN entry applied to the interface with index ifIndexNext;

or:

2.mplsFTNMapStorageType.firstIfIndex.0.p, if there are no more entries in mplsFTNMapTable, where firstIfIndex is the first entry in ifTable to which an FTN entry has been mapped.

Use the above steps to retrieve all the applied FTN entries on a per-interface basis in application order. Note that the number of retrieval operations is the same as the number of applied FTN entries (i.e., the minimum number of GETNEXT operations needed using any indexing scheme).

Agents MUST NOT allow the same FTN entry as specified by mplsFTNMapCurrIndex to be applied multiple times to the same interface.

Agents MUST NOT allow the creation of rows in this table until the corresponding rows are created in the mplsFTNTable.

If a row in mplsFTNTable is destroyed, the agent MUST destroy the corresponding entries (i.e., ones with a matching value of mplsFTNCurrIndex) in this table as well."

```
::= { mplsFTNObjects 5 }
```

```
mplsFTNMapEntry OBJECT-TYPE
```

```
SYNTAX MplsFTNMapEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Each conceptual row represents the application of an FTN rule at a specific position in the list of FTN rules applied on an interface. "
```

```
INDEX {
    mplsFTNMapIndex,
    mplsFTNMapPrevIndex,
    mplsFTNMapCurrIndex
}
```

```
::= { mplsFTNMapTable 1 }
```

```
MplsFTNMapEntry ::= SEQUENCE {
```

```

    mplsFTNMapIndex      InterfaceIndexOrZero,
    mplsFTNMapPrevIndex  MplsFTNEntryIndexOrZero,
    mplsFTNMapCurrIndex  MplsFTNEntryIndex,
    mplsFTNMapRowStatus  RowStatus,
    mplsFTNMapStorageType StorageType
}

mplsFTNMapIndex OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The interface index that this FTN entry is being
        applied to. A value of zero indicates an entry that is
        applied all interfaces.

        Entries mapped to an interface by specifying its (non-
        zero) interface index in mplsFTNMapIndex are applied
        ahead of entries with mplsFTNMapIndex equal to zero."
    ::= { mplsFTNMapEntry 1 }

mplsFTNMapPrevIndex OBJECT-TYPE
    SYNTAX      MplsFTNEntryIndexOrZero
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index of the previous FTN entry that was applied to
        this interface. The special value zero indicates that
        this should be the first FTN entry in the list."
    ::= { mplsFTNMapEntry 2 }

mplsFTNMapCurrIndex OBJECT-TYPE
    SYNTAX      MplsFTNEntryIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Index of the current FTN entry that is being applied to
        this interface."
    ::= { mplsFTNMapEntry 3 }

mplsFTNMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus {
                    active(1),
                    createAndGo(4),
                    destroy(6)
                }
    MAX-ACCESS  read-create
    STATUS      current

```

DESCRIPTION

"Used for controlling the creation and deletion of this row.

All writable objects in this row may be modified at any time.

If a conceptual row in mplsFTNMapTable points to a conceptual row in mplsFTNTable which is subsequently deleted, the corresponding conceptual row in mplsFTNMapTable MUST also be deleted by the agent."

```
::= { mplsFTNMapEntry 4 }
```

```
mplsFTNMapStorageType OBJECT-TYPE
```

```
SYNTAX StorageType
```

```
MAX-ACCESS read-create
```

```
STATUS current
```

DESCRIPTION

"The storage type for this entry. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in this row."

```
DEFVAL { nonVolatile }
```

```
::= { mplsFTNMapEntry 5 }
```

```
-- End of mplsFTNMapTable
```

```
-- FTN entry performance table
```

```
mplsFTNPerfTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF MplsFTNPerfEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS current
```

DESCRIPTION

"This table contains performance statistics on FTN entries on a per-interface basis."

```
::= { mplsFTNObjects 6 }
```

```
mplsFTNPerfEntry OBJECT-TYPE
```

```
SYNTAX MplsFTNPerfEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS current
```

DESCRIPTION

"Each entry contains performance information for the specified interface and an FTN entry mapped to this interface."

```
INDEX { mplsFTNPerfIndex, mplsFTNPerfCurrIndex }
```

```
::= { mplsFTNPerfTable 1 }
```

```

MplsFTNPerfEntry ::= SEQUENCE {
    mplsFTNPerfIndex          InterfaceIndexOrZero,
    mplsFTNPerfCurrIndex     MplsFTNEntryIndex,
    mplsFTNPerfMatchedPackets Counter64,
    mplsFTNPerfMatchedOctets Counter64,
    mplsFTNPerfDiscontinuityTime TimeStamp
}

```

```

mplsFTNPerfIndex OBJECT-TYPE
    SYNTAX          InterfaceIndexOrZero
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The interface index of an interface that an FTN entry
        has been applied/mapped to. Each instance of this
        object corresponds to an instance of mplsFTNMapIndex."
    ::= { mplsFTNPerfEntry 1 }

```

```

mplsFTNPerfCurrIndex OBJECT-TYPE
    SYNTAX          MplsFTNEntryIndex
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Index of an FTN entry that has been applied/mapped to
        the specified interface. Each instance of this object
        corresponds to an instance of mplsFTNMapCurrIndex."
    ::= { mplsFTNPerfEntry 2 }

```

```

mplsFTNPerfMatchedPackets OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Number of packets that matched the specified FTN entry
        if it is applied/mapped to the specified interface.
        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        mplsFTNDiscontinuityTime."
    ::= { mplsFTNPerfEntry 3 }

```

```

mplsFTNPerfMatchedOctets OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Number of octets that matched the specified FTN entry
        if it is applied/mapped to the specified interface."

```

```

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        mplsFTNDiscontinuityTime."
 ::= { mplsFTNPerfEntry 4 }

mplsFTNPerfDiscontinuityTime 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 entry's counters 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."
 ::= { mplsFTNPerfEntry 5 }

-- End of mplsFTNPerfTable

-- Module compliance.

-- Top level object IDs.

mplsFTNGroups
    OBJECT IDENTIFIER ::= { mplsFTNConformance 1 }
mplsFTNCompliances
    OBJECT IDENTIFIER ::= { mplsFTNConformance 2 }

-- Compliance requirement for fully compliant implementations.
mplsFTNModuleFullCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "Compliance statement for agents that provide full
        support for MPLS-FTN-STD-MIB."

MODULE IF-MIB -- The Interfaces Group MIB, RFC 2863.
MANDATORY-GROUPS {
    ifGeneralInformationGroup,
    ifCounterDiscontinuityGroup
}

MODULE -- This module.
MANDATORY-GROUPS {
    mplsFTNRuleGroup,
    mplsFTNMapGroup,
    mplsFTNPerfGroup
}

```



```

}
```

```
OBJECT mplsFTNAddrType
```

```
SYNTAX InetAddressType { ipv4(1), ipv6(2) }
```

```
DESCRIPTION
```

```

    "An implementation is only required to support IPv4
    and/or IPv6 addresses. An implementation is only
    required to support the address types that are actually
    supported on the LSR."
```

```
OBJECT mplsFTNSourceAddrMin
```

```
SYNTAX      InetAddress (SIZE (4 | 20))
```

```
DESCRIPTION
```

```

    "An implementation is only required to support IPv4
    and/or IPv6 addresses. An implementation is only
    required to support the address types that are actually
    supported on the LSR."
```

```
OBJECT mplsFTNSourceAddrMax
```

```
SYNTAX      InetAddress (SIZE (4 | 20))
```

```
DESCRIPTION
```

```

    "An implementation is only required to support IPv4
    and/or IPv6 addresses. An implementation is only
    required to support the address types that are actually
    supported on the LSR."
```

```
OBJECT mplsFTNDestAddrMin
```

```
SYNTAX      InetAddress (SIZE (4 | 20))
```

```
DESCRIPTION
```

```

    "An implementation is only required to support IPv4
    and/or IPv6 addresses. An implementation is only
    required to support the address types that are actually
    supported on the LSR."
```

```
OBJECT mplsFTNDestAddrMax
```

```
SYNTAX      InetAddress (SIZE (4 | 20))
```

```
DESCRIPTION
```

```

    "An implementation is only required to support IPv4
    and/or IPv6 addresses. An implementation is only
    required to support the address types that are actually
    supported on the LSR."
```

```
 ::= { mplsFTNCompliances 1 }
```

```
-- Compliance requirement for read-only implementations.
```

```
mplsFTNModuleReadOnlyCompliance MODULE-COMPLIANCE
```

```
STATUS current
```

```
DESCRIPTION
```

```

    "Compliance requirement for implementations that only
```

provide read-only support for MPLS-FTN-STD-MIB. Such devices can then be monitored but cannot be configured using this MIB module."

MODULE IF-MIB -- The interfaces Group MIB, RFC 2863

```
MANDATORY-GROUPS {
    ifGeneralInformationGroup,
    ifCounterDiscontinuityGroup
}
```

MODULE -- This module

```
MANDATORY-GROUPS {
    mplsFTNRuleGroup,
    mplsFTNMapGroup,
    mplsFTNPerfGroup
}
```

OBJECT mplsFTNIndexNext

MIN-ACCESS not-accessible

DESCRIPTION

"This object is not needed when mplsFTNTable is implemented as read-only."

OBJECT mplsFTNRowStatus

SYNTAX RowStatus { active(1) }

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required, and active is the only status that needs to be supported."

OBJECT mplsFTNDescr

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required."

OBJECT mplsFTNMask

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required."

OBJECT mplsFTNAddrType

SYNTAX InetAddressType { ipv4(1), ipv6(2) }

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required. An implementation is only required to support IPv4 and IPv6 addresses."

OBJECT mplsFTNSourceAddrMin

```
SYNTAX      InetAddress (SIZE (4 | 20))
MIN-ACCESS  read-only
DESCRIPTION
    "Write access is not required. An implementation is only
    required to support IPv4 and IPv6 addresses."

OBJECT mplsFTNSourceAddrMax
SYNTAX      InetAddress (SIZE (4 | 20))
MIN-ACCESS  read-only
DESCRIPTION
    "Write access is not required. An implementation is only
    required to support IPv4 and IPv6 addresses."

OBJECT mplsFTNDestAddrMin
SYNTAX      InetAddress (SIZE (4 | 20))
MIN-ACCESS  read-only
DESCRIPTION
    "Write access is not required. An implementation is only
    required to support IPv4 and IPv6 addresses."

OBJECT mplsFTNDestAddrMax
SYNTAX      InetAddress (SIZE (4 | 20))
MIN-ACCESS  read-only
DESCRIPTION
    "Write access is not required. An implementation is only
    required to support IPv4 and IPv6 addresses."

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

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

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

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

OBJECT mplsFTNProtocol
```

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

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

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

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

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

OBJECT mplsFTNMapRowStatus
SYNTAX        RowStatus { active(1) }
MIN-ACCESS    read-only
DESCRIPTION
    "Write access is not required, and active(1) is the only
    status that needs to be supported."

OBJECT mplsFTNMapStorageType
MIN-ACCESS    read-only
DESCRIPTION
    "Write access is not required."
 ::= { mplsFTNCompliances 2 }

-- Units of conformance.
mplsFTNRuleGroup OBJECT-GROUP
OBJECTS {
    mplsFTNIndexNext,
    mplsFTNTableLastChanged,
    mplsFTNRowStatus,
    mplsFTNDescr,
    mplsFTNMask,
    mplsFTNAddrType,
    mplsFTNSourceAddrMin,
    mplsFTNSourceAddrMax,
```

```
        mplsFTNDestAddrMin,
        mplsFTNDestAddrMax,
        mplsFTNSourcePortMin,
        mplsFTNSourcePortMax,
        mplsFTNDestPortMin,
        mplsFTNDestPortMax,
        mplsFTNProtocol,
        mplsFTNActionType,
        mplsFTNActionPointer,
        mplsFTNDscp,
        mplsFTNStorageType
    }
    STATUS current
    DESCRIPTION
        "Collection of objects that implement MPLS FTN rules."
    ::= { mplsFTNGroups 1 }

mplsFTNMapGroup OBJECT-GROUP
    OBJECTS {
        mplsFTNMapTableLastChanged,
        mplsFTNMapRowStatus,
        mplsFTNMapStorageType
    }
    STATUS current
    DESCRIPTION
        "Collection of objects that implement activation of MPLS
        FTN entries on interfaces."
    ::= { mplsFTNGroups 2 }

mplsFTNPerfGroup OBJECT-GROUP
    OBJECTS {
        mplsFTNPerfMatchedPackets,
        mplsFTNPerfMatchedOctets,
        mplsFTNPerfDiscontinuityTime
    }
    STATUS current
    DESCRIPTION
        "Collection of objects providing MPLS FTN performance
        information."
    ::= { mplsFTNGroups 3 }

END
```

10. Security Considerations

This MIB module can be used to configure LSRs to redirect non-MPLS traffic into an MPLS cloud. As such, improper manipulation of the objects represented in this MIB module may result in traffic being redirected to unintended destinations, potentially resulting in denial of service to end-users.

There are a number of management objects defined in this MIB module with 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. These are the tables and objects and their sensitivity/vulnerability:

- `mplsFTNTable` and `mplsFTNMapTable` can be used to create packet matching rules for classifying IPv4 or IPv6 traffic and redirecting matched packets into the MPLS cloud. Modifying objects in these tables can result in the misdirection of traffic and potential denial of service to end-users. It may also result in traffic which was intended to be redirected into the MPLS cloud being routed through the IP network instead, potentially resulting in degradation of service quality or outright denial of service.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- `mplsFTNPerfTable` provides counters for monitoring the performance of packet classification rules defined in `mplsFTNTable` and `mplsFTNMapTable`. Unauthorized read access to objects in these tables may be used to gain traffic flow information.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), even then, 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 module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED that SNMPv3 be deployed and cryptographic security be enabled. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects to only those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

11. IANA Considerations

As described in [MPLSMGMT] and as requested in [RFC3811], MPLS related standards-track MIB modules should be rooted under the mplsStdMIB subtree. New assignments can only be made by a standards action as specified in [RFC2434].

11.1. IANA Considerations for MPLS-FTN-STD-MIB

The IANA has assigned mplsStdMIB 8 to the MPLS-FTN-STD-MIB module specified in this document.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.
- [RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, May 2002.

- [RFC3291] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 3291, May 2002.
- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.
- [RFC3813] Srinivasan, C., Viswanathan, A., and T. Nadeau, "Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base (MIB)", RFC 3813, June 2004.
- [RFC3811] Nadeau, T., and J. Cucchiara, J., Editors, "Definition of Textual Conventions (TCs) for Multi-Protocol Label Switching (MPLS) Management", RFC 3811, June 2004.
- [RFC3812] Srinivasan, C., Viswanathan, A., and T. Nadeau, "Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Management Information Base (MIB)", RFC 3812, June 2004.

12.2. Informative References

- [MPLSMGMT] Nadeau, T., Srinivasan, C., and A. Farrel, "Multiprotocol Label Switching (MPLS) Management Overview", Work in Progress, September 2003.
- [RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC1519] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, September 1993.
- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,
"Definition of the Differentiated Services Field (DS
Field) in the IPv4 and IPv6 Headers", RFC 2474, December
1998.
- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart,
"Introduction and Applicability Statements for Internet-
Standard Management Framework", RFC 3410, December 2002.

13. Acknowledgements

We would particularly like to thank Bert Wijnen for the substantial time and effort he spent in helping us improve this document. We would also like to thank David Perkins, Joan Cucchiara, Mike Piecuch, and Adrien Grise for their insightful comments and additions to this document.

14. Authors' Addresses

Thomas D. Nadeau
Cisco Systems, Inc.
300 Apollo Drive
Chelmsford, MA 01824

Phone: +1-978-244-3051
EMail: tnadeau@cisco.com

Cheenu Srinivasan
Bloomberg L.P.
499 Park Avenue
New York, NY 10022

Phone: +1-212-893-3682
EMail: cheenu@bloomberg.net

Arun Viswanathan
Forcel0 Networks, Inc.
1440 McCarthy Blvd
Milpitas, CA 95035

Phone: +1-408-571-3516
EMail: arunv@forcel0networks.com

15. Full Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.