The ATM Forum
Technical Committee

Integrated Local Management Interface (ILMI) Specification Version 4.0

af-ilmi-0065.000

September, 1996

Contents

Scope

This document specifies how the Simple Network Management Protocol (SNMP) and an ATM Interface Management Information Base (MIB) are used to provide any ATM device (e.g. End-systems, Switches, etc.) with status and configuration information concerning the Virtual Path Connections, Virtual Channel Connections, registered ATM Network Prefixes, registered ATM Addresses, registered services, and capabilities available at its ATM Interfaces.

> Note: At the time the original version of this specification was developed, it was expected that these procedures would be used for only a short period of time. The specification was titled *Interim* to emphasize this expectation. These expectations have not come to pass. It is now expected that these procedures will be used *indefinitely*.

The Integrated Local Management Interface (ILMI) fits into the overall model for an ATM device as illustrated in Figure 1 as clarified by the following principles and options.

- Each ATM device (i.e. switches, end-systems, etc.) shall support one or more ATM Interfaces.
- ILMI functions for an ATM Interface provide configuration, status, and control information about physical and ATM layer parameters of the ATM Interface.
- There is a per-ATM Interface set of managed objects, the ATM Interface ILMI attributes, that is sufficient to support the ILMI functions for each ATM Interface.
- The ATM Interface ILMI attributes are organized in a standard MIB structure; there is one MIB structure instance for each ATM Interface on each ATM device.
- For any ATM device, there is an ATM Interface Management Entity (IME) associated with each ATM Interface that supports the ILMI functions for that ATM Interface.
- When two ATM devices are connected across an (point-to-point) ATM Interface, there are two ATM Interface Management Entities (IMEs) associated with that ATM Interface, one IME for each ATM device, and two such IMEs are defined as adjacent IMEs.
- The ILMI communication takes place between adjacent IMEs over physical links or virtual links (i.e. Virtual Path Connections used by IISP or PNNI).
- The ILMI communication protocol is an open protocol (i.e. SNMP/AAL5).
- An ATM Interface Management Entity (IME) can access, via the ILMI communication protocol, the ATM Interface MIB information associated with its adjacent IME.
- Whether access to additional information (beyond the adjacent IME's ATM Interface MIB information) is available via the ILMI communication protocol is currently unspecified, and is regarded as a vendor implementation choice.
- This document pertains to the ATM Interface MIB structure of the "Local ATM Interface" (i.e. between adjacent IMEs) only.

- ILMI functions for an ATM User Network Interface (UNI) also provide for address registration across the UNI.
- ILMI functions for LAN Emulation provide for auto-configuration of a LAN Emulation Client (LEC). Further details on LEC auto-configuration are contained in [LANE1.0].
- Future ATM Forum ILMI extensions will be defined in new revisions of this document.

SNMP, without UDP and IP addressing, along with an ATM Interface MIB were chosen for the ILMI.

> Note: Although [IISP1.0] specifically states that ILMI is not used over IISP 1.0 links, this specification provides an update. ILMI 4.0 may be run over IISP links, with the exception of the address registration procedures.

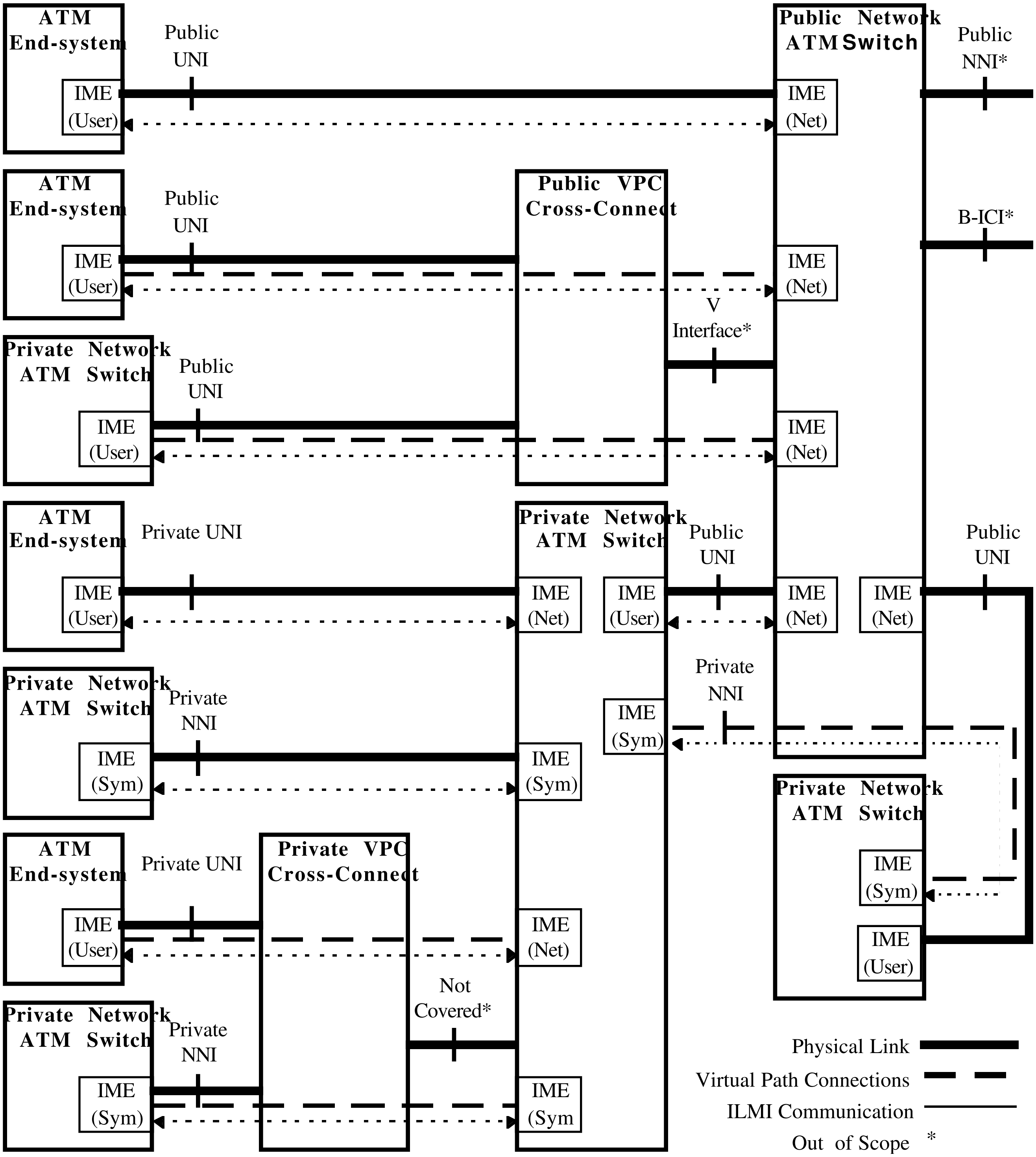


Figure 1 - Definition and Context of ILMI

1. ILMI Functions

The ILMI supports bi-directional exchange of ATM Interface parameters between two connected ATM Interface Management Entities (IMEs). Each IME contains an agent application and a management application. Unless otherwise stated for specific portions of the ILMI, every IME contains the same ATM Interface MIB. However, semantics of some MIB objects may be interpreted differently depending on the role of an individual IME. An example list of the equipment that will use the ILMI include:

- Workstations and computers with ATM interfaces which send their data in ATM cells across an ATM Interface to an ATM switch.
- ATM Network switches which send ATM cells across an ATM Interface to other ATM devices.
- Higher layer switches such as internet routers, frame relay switches, or LAN bridges, that transfer their frames within ATM cells and forward the cells across an ATM Interface to an ATM switch.

This document describes four ATM Interface MIB modules: the Textual Conventions MIB, the Link Management MIB, the Address Registration MIB, and the Service Registry MIB. The Textual Conventions MIB defines a number of common Textual Conventions and Object Identifiers in a single module so that other MIB modules may import them in a mutually consistent fashion. The Link Management MIB provides a general-purpose link management facility for all ATM Interfaces. The Address Registration MIB provides a mechanism for the registration of ATM Addresses for an User Network Interface. The Service Registry MIB provides a general-purpose service registry for locating ATM network services such as the LAN Emulation Configuration Server (LECS).

All future MIB modules created by the ATM Forum Technical Committee as extensions to the ILMI will be incorporated in revisions of this specification.

2. ILMI Service Interface

The ILMI uses SNMP for management and control operations of information across the ATM Interface. The ATM Interface information will be represented in a Management Information Base. The types of information that will be available in the ATM Interface MIB are as follows:

- Physical Layer
- ATM Layer
- Virtual Path (VP) Connections
- Virtual Channel (VC) Connections
- Address Registration Information
- Service Registry

The tree structure of the ATM Interface MIB is depicted in Figure 2.

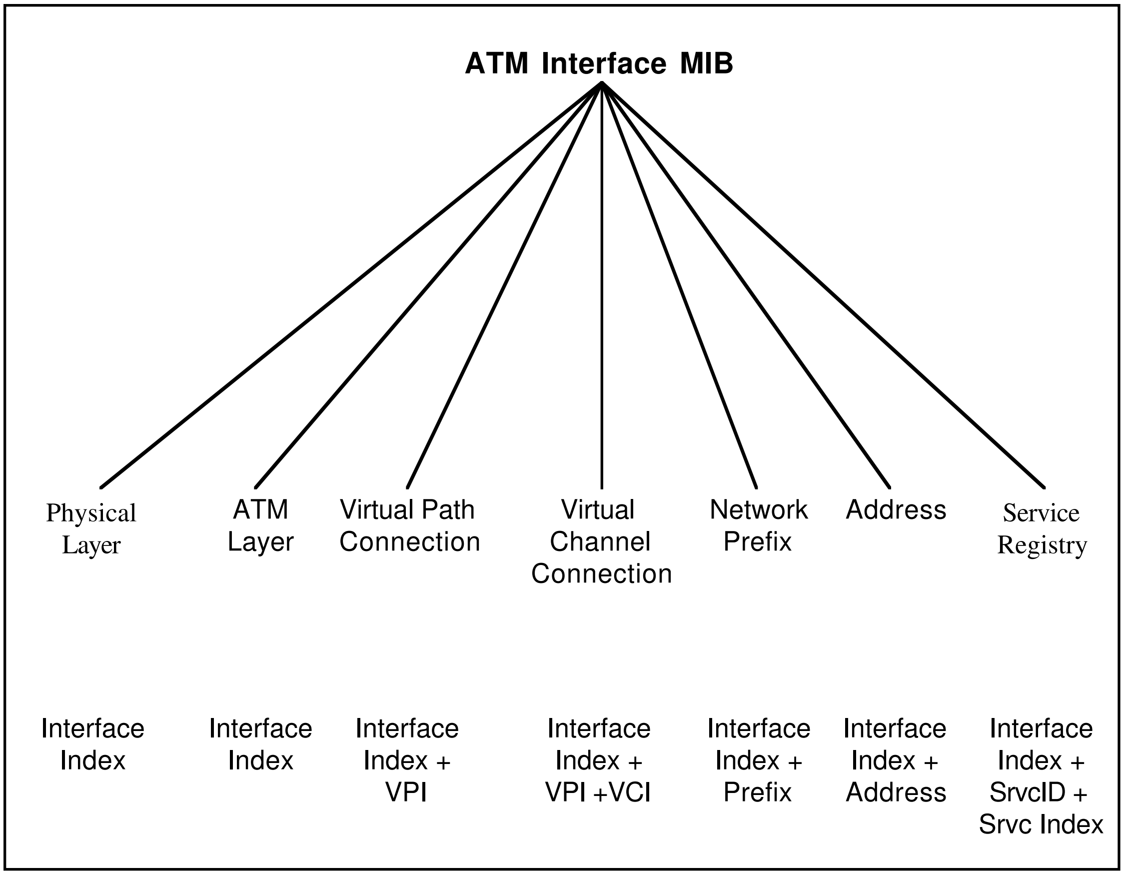


Figure 2 - ATM Interface MIB Tree Structure

The ATM Interface MIB may be extended over time to allow for the addition of new items without requiring any changes to the management protocol or framework. In addition, vendors can define private ATM Interface MIB extensions to support additional or proprietary features of their products.

The following sections introduce the groups which categorize the management information. An entire tree group is either Optional **(O)**, Conditionally Required **(CR)**, Required **(R)**, or Obsolete/Deprecated **(D)**. If a group is Required, then every element in the group is Required. For a group which is Conditionally Required it follows that every element in the group is required if implemented.

2.1 System (R)

The System group from RFC 1213 [MIB-II] is included to support the objects as defined in section 6.

2.2 Physical Layer (R)

The ILMI provides access to management information identifying the Physical Layer interface.

When ILMI communication takes place over a physical link, there is one Physical Layer group for that physical interface. When ILMI communication takes place over a virtual link (i.e. a Virtual Path Connection used by PNNI), the physical layer management information is present and represents the virtual interface.

Each physical or virtual interface has a set of specific attributes and information associated with it.

The Physical Port Group specified in versions 3.1 and older of the ILMI Specification provided configuration information and statistics about the ATM Interface's Physical Layer interface. Most of these objects have now been deprecated. This information is available via standard Network Management MIBs.

2.3 ATM Layer (R)

The ILMI provides access to management information about the ATM Layer .

There is one ATM layer group per physical or virtual interface.

Certain attributes of the ATM layer are common across all Virtual Path Connections (VPCs) and Virtual Channel Connections (VCCs) at this ATM Interface.

Configuration information at the ATM layer relates to the size of the VPI and VCI address fields in the ATM cell header, number of configured permanent VPCs and permanent VCCs, and maximum number of VPCs and VCCs allowed at this ATM Interface.

2.4 ATM Layer Statistics (D)

The ATM Layer Statistics Group specified in versions 3.1 and older of the User-Network Interface Specification provided statistics for the ATM layer. This group has now been deprecated. This information is now available via standard Network Management MIBs.

2.5 Virtual Path Connections (R)

In the context of supporting ILMI functions, a point-to-point Virtual Path Connection (VPC) extends between two ATM Interfaces that terminate the VPC.

On the local ATM Layer interface the VPC is uniquely identified by the VPI value, when ILMI communication takes place over a physical link.

The status information indicates the IME's knowledge of the VPC status.

Configuration information relates to the QoS parameters for the VPC local end point.

2.6 Virtual Channel Connections (R)

In the context of supporting ILMI functions, a point-to-point Virtual Channel Connection (VCC) extends between two ATM Interfaces that terminate the VCC.

On the local ATM Layer interface the VCC is uniquely identified by the VPI and VCI value, when ILMI communication takes place over a physical link. When ILMI communication takes place over a virtual link (i.e. a Virtual Path Connection used by PNNI), the VPI value is not used. In this case, the VPI value should transmitted as zero and ignored upon reception.

The status information indicates the IME's knowledge of the VCC status.

Configuration information relates to the QoS parameters for the VCC local end point.

2.7 Network Prefix (CR)

The ILMI provides an Address Registration mechanism which allows switches to automatically configure Network Prefixes in end-systems.

2.8 Address (CR)

The ILMI provides an Address Registration mechanism which allows end-systems to automatically configure ATM Addresses for ATM Interfaces on switches.

2.9 Service Registry (O)

The ILMI provides a general-purpose Service Registry for locating ATM network services such as the Lan Emulation Configuration Server (LECS).

3. Simple Network Management Protocol (SNMP)

The SNMP protocol as defined in RFC 1157 consists of four types of operations which are used to manipulate management information. These are:

1.	Get	Used to retrieve specific management information.
2.	Get-Next	Used to retrieve, via traversal of the MIB, management information.
3.	Set	Used to alter management information.
4.	Trap	Used to report extraordinary events.

These four operations are implemented using five types of PDUs:

1.	GetRequest-PDU	Used to request a Get operation.
2.	GetNextRequest-PDU	Used to request a Get-Next operation.
3.	GetResponse-PDU	Used to respond to a Get, Get-Next, or Set operation.
4.	SetRequest-PDU	Used to request a Set operation.
5.	Trap-PDU	Used to report a Trap operation.

4. Objects

4.1 Model for Managed Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Information related to the operation of the ATM Interface is organized into a MIB in a hierarchical fashion as defined in section 2. Each ATM Interface corresponds to just one physical interface. Logically, the ATM Interface MIB accessed over the ILMI corresponds to the single ATM Interface/physical interface. Implementations of devices which have multiple physical interfaces (e.g., multiple interface end user devices, private switches and public switches) may implement a single ATM Interface MIB, indexed for each ATM Interface/physical interface.

The ATM Interface MIB attributes are used in the standard fashion: all attributes are by default read only across the ILMI, unless stated otherwise as readable and writeable across the ILMI for a specific MIB variable.

The ILMI MIB modules specified herein are organized under the ATM Forum's node under the enterprises node of the standard SMI as defined in RFC 1212. This means it is prefixed by the following tree path or name, 1.3.6.1.4.1.353.

4.2 Structure of Management Information

Objects in the ATM Interface MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) [ASN.1] defined by the Structure of Management Information (SMI) [SMI]. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. 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 OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [ASN.1-BER], subject to the additional requirements imposed by the SNMP [SNMP].

4.3 Textual Conventions

Several data types are used as textual conventions in this document. These textual conventions have no effect on either the syntax nor the semantics of any managed object. Objects defined using these conventions are always encoded by means of the rules that define their primitive type. Hence, no changes to the SMI or the SNMP are necessary to accommodate these textual conventions which are adopted merely for the convenience of readers.

4.4 Use of Counters

The ATM Interface MIB defines all counter values using the standard SMI data type: Counter. This data type is defined [SMI] as representing a non-negative integer which monotonically increases until it reaches a maximum value of 2^32-1 (4294967295 decimal), when it wraps around and starts increasing again from zero.

This definition disallows the clearing of Counter values, in order to prevent interference which would otherwise occur if two network managers were accessing the counters concurrently. Instead, interval values are obtained as the delta between a Counter’s values at the beginning and end of a period.

4.5 Meaning of Transmit/Receive

The terms ‘transmit’ and ‘receive’ are used in the following MIB modules’ object descriptors and textual descriptions. In each case, these terms are used from the perspective of the device local to the management information being defined.

5. ILMI Protocol

The use of SNMP messages will be according to the following requirements.

5.1 Use of VCCs

One VCC will be used for sending AAL-encapsulated SNMP messages between adjacent IMEs. This VCC is used for requests, responses, and traps, differentiated according to the SNMP PDU type.

(R) Encapsulation of ILMI SNMP messages in AAL5 shall be accomplished as specified in [AAL5]. The following procedures shall be used regarding AAL5 Common Part Convergence Sublayer (CPCS) service primitives and parameters:

- The Service Specific Convergence Sublayer shall be null,
- The message mode is used,
- The corrupted data delivery option is not used,
- The SNMP Message is carried in the Interface Data (ID) parameter of CPCS-UNITDATA.invoke and CPCS-UNITDATA.signal,
- The CPCS-Loss-Priority (CPCS-LP) parameter is set to “0” at the transmitter and ignored at the receiver,
- The CPCS-Congestion-Indication (CPCS-CI) parameter is set to “0” at the transmitter and ignored at the receiver,
- The CSCS-User-to-User-Indication (CPCS-UU) parameter is set to “0” at the transmitter and ignored at the receiver.

(R) At all times one VCC will be provisioned for the ILMI. The default value for provisioning this VCC is VPI=0, VCI=16, however, the VPI/VCI value must be configurable.

(R) The cells carrying ILMI messages shall be sent with cell loss priority (CLP = 0)

(R) The throughput of SNMP traffic on the ILMI VCC should be no more than approximately one percent of the link bandwidth.

5.2 Message Format (R)

The message format specified in [SNMP] will be used. That is, messages will be formatted according to SNMP version 1, not SNMP version 2. Any use of SNMP version 2 is for future study.

All SNMP messages will use the community name “ILMI”, that is, the OCTET STRING value: ‘494C4D49’h.

In all SNMP Traps, the agent-addr field (which has syntax NetworkAddress), will always have the IpAddress value: 0.0.0.0.

In all SNMP Traps, the time-stamp field in the Trap-PDU will contain the value of the IME's sysUpTime MIB object at the time of trap generation (sysUpTime is defined in the system group of MIB-II).

In any standard SNMP Traps, (e.g., the coldStart Trap), the enterprise field in the Trap-PDU will contain the value of the agent's sysObjectID MIB object (sysObjectID is defined in the system group of MIB-II).

The supported traps are coldStart and enterpriseSpecific. Other traps not explicitly specified in this document are not supported, should not be sent, and must be ignored upon reception.

5.3 Message Sizes (R)

All ILMI implementations will be able to accept SNMP messages of size up to and including 484 octets. Larger messages should not be sent unless the originator has information (e.g., via some out-of-band mechanism) that the receiver supports larger messages.

5.4 ILMI Traffic Requirements

The traffic requirements relating to the ATM connection used for ILMI communication are as follows:

(R) The VCC used for ILMI communication shall support a sustainable cell rate, R(s), no more than 1% of the ATM Interface physical line rate and a peak cell rate, R(p), no more than 5% of the ATM Interface physical line rate.

(R) The ILMI traffic burst length, L(b), shall be no more than 484 octets.

(R) The ILMI traffic bursts inter-arrival time, T(b), should be greater than or equal to (L(b)/(R(p) x 1%)), where L(b) and R(p) are respectively the ILMI burst length and the ILMI traffic peak rate.

5.5 Message Response Time

Response time refers to the elapsed time from the submission of an SNMP message (e.g., GetRequest, GetNextRequest, or SetRequest message) by an IME across an ATM Interface to the receipt of the corresponding SNMP message (e.g., GetResponse message) from the adjacent IME. An SNMP GetRequest, GetNextRequest, or SetRequest message is defined in this context as a request concerning a single object. The following specifies ILMI response time requirements.

(R) The IME should support maximum Response Times of 1 second for 95% of all SNMP GetRequests, GetNextRequests or SetRequests containing a single object received from an adjacent IME independent of the ATM Interface's physical line rate.

5.6 Object Value Data Currentness

Data currentness refers to the maximum elapsed time since an object value in the ATM Interface MIB was known to be current. The following specifies the requirements on the Data Currentness of the ILMI objects and the event notifications.

(R) The ATM Interface MIB objects should have the Data Currentness of a maximum of 30 seconds.

(R) The IME should support event notifications (i.e., SNMP Traps) for generic SNMP events (e.g., coldStart) within 2 seconds of the event detection by the IME.

6. Relationship to Other MIBs

It is required that an ATM device which implements the ATM Interface MIB will also implement the ‘system’ group defined in MIB-II. An IME must provide access to the ‘system’ group via the ILMI communications protocol.

6.1 Relationship to the ‘system’ group (R)

In MIB-II, the ‘system’ group is defined as being mandatory for all systems such that each managed entity contains one instance of each object in the ‘system’ group. Thus, those objects apply to the entity even if the entity’s sole functionality is the forwarding of ATM cells.

RFC 1213 is the authoritative source for the definition of objects in the ‘system’ group. The group consists of the following 7 objects:

(For each textual object for which the device is not configured with a value, the object’s value is a string of length 0.)

sysDescr “A textual description of the entity. This value should include the full name and version identification of the system’s hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters.”

sysObjectID “The vendor’s authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining ‘what kind of box’ is being managed. For example, if vendor ‘Flintstones, Inc.’ was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its ‘Fred Router’.”

sysUpTime “The time (in hundredths of a second) since the IME was last re-initialized.”

sysContact “The textual identification of the contact person for this managed node, together with information on how to contact this person.”

sysName “An administratively-assigned textual name for this managed node.”

sysLocation “A textual description of the physical location of this device (e.g., `telephone closet, 3rd floor’).”

sysServices “A value which indicates the set of services that this entity primarily offers. The value is a sum of individual values, each representing a particular switching function. The layers are:

layer	**functionality**
1	physical (e.g. repeaters)
2	datalink/subnetwork (e.g. ATM switches, IEEE 802.1 bridges)
3	internet (e.g. IP routers)
4	end-to-end (e.g. IP hosts)
7	applications (e.g. mail relays).”

7. Textual Conventions MIB

The Textual Conventions MIB defines a number of common Textual Conventions and Object Identifiers in a single module so that other MIB modules may import them in a mutually consistent fashion.

7.1 MIB Definitions

```
ATM-FORUM-TC-MIB DEFINITIONS ::= BEGIN

IMPORTS
        enterprises                     FROM RFC1155-SMI;


--              Textual Conventions

-- Boolean values use this data type from RFC-1903, "Textual Conventions
-- for Version 2 of the Simple Network Management Protocol (SNMPv2)"
TruthValue ::= INTEGER { true(1), false(2) }

-- CLNP address values use this data type from RFC-1238, "CLNS MIB for
-- use with Connectionless Network Protocol (ISO 8473) and End System
-- to Intermediate System (ISO 9542)"
ClnpAddress ::= OCTET STRING (SIZE (1..21))

-- ATM Service Categories use this data type (See [TM4.0]):
AtmServiceCategory ::=
        INTEGER {
                other(1),
                cbr(2),
                rtVbr(3),
                nrtVbr(4),
                abr(5),
                ubr(6)
        }

-- ATM End-System Addresses use this data type:
AtmAddress ::= OCTET STRING (SIZE (8 | 20))

-- Network-Prefixes for an ATM Address use this data type:
NetPrefix ::= OCTET STRING (SIZE (8 | 13))

-- In both the AtmAddress and NetPrefix conventions, Native E.164 addresses
-- are represented as 8 octets using the format specified in section
-- 3.1.1.3 of the ATM Forum UNI Signalling 4.0 specification.
-- In contrast, an NSAP-encoded address is 20 octets, and an NSAP-encoded
-- network prefix is 13 octets long.
```

```
--              MIB Groups

-- a subtree for defining ATM Forum MIB object types
atmForum                OBJECT IDENTIFIER ::= { enterprises 353 }

-- a subtree for defining administrative object types
atmForumAdmin           OBJECT IDENTIFIER ::= { atmForum 1 }
atmfTransmissionTypes   OBJECT IDENTIFIER ::= { atmForumAdmin 2 }
atmfMediaTypes          OBJECT IDENTIFIER ::= { atmForumAdmin 3 }
atmfTrafficDescrTypes   OBJECT IDENTIFIER ::= { atmForumAdmin 4 }
atmfSrvcRegTypes        OBJECT IDENTIFIER ::= { atmForumAdmin 5 }

-- a subtree for defining ATM Interface MIB object types
atmForumUni             OBJECT IDENTIFIER ::= { atmForum 2 }
atmfPhysicalGroup       OBJECT IDENTIFIER ::= { atmForumUni 1 }
atmfAtmLayerGroup       OBJECT IDENTIFIER ::= { atmForumUni 2 }
atmfAtmStatsGroup       OBJECT IDENTIFIER ::= { atmForumUni 3 }
atmfVpcGroup            OBJECT IDENTIFIER ::= { atmForumUni 4 }
atmfVccGroup            OBJECT IDENTIFIER ::= { atmForumUni 5 }
atmfAddressGroup        OBJECT IDENTIFIER ::= { atmForumUni 6 }
atmfNetPrefixGroup      OBJECT IDENTIFIER ::= { atmForumUni 7 }
atmfSrvcRegistryGroup   OBJECT IDENTIFIER ::= { atmForumUni 8 }
atmfVpcAbrGroup         OBJECT IDENTIFIER ::= { atmForumUni 9 }
atmfVccAbrGroup         OBJECT IDENTIFIER ::= { atmForumUni 10 }
atmfAddressRegistrationAdminGroup OBJECT IDENTIFIER ::= { atmForumUni 11 }


--              Object Identifier definitions

-- Transmission Types: These values are no longer used
atmfUnknownType         OBJECT IDENTIFIER ::= { atmfTransmissionTypes 1}
atmfSonetSTS3c          OBJECT IDENTIFIER ::= { atmfTransmissionTypes 2 }
atmfDs3                 OBJECT IDENTIFIER ::= { atmfTransmissionTypes 3 }
atmf4B5B                OBJECT IDENTIFIER ::= { atmfTransmissionTypes 4 }
atmf8B10B               OBJECT IDENTIFIER ::= { atmfTransmissionTypes 5 }
atmfSonetSTS12c         OBJECT IDENTIFIER ::= { atmfTransmissionTypes 6 }
atmfE3                  OBJECT IDENTIFIER ::= { atmfTransmissionTypes 7 }
atmfT1                  OBJECT IDENTIFIER ::= { atmfTransmissionTypes 8 }
atmfE1                  OBJECT IDENTIFIER ::= { atmfTransmissionTypes 9 }

-- Media Types: These values are no longer used
atmfMediaUnknownType    OBJECT IDENTIFIER ::= { atmfMediaTypes 1 }
atmfMediaCoaxCable      OBJECT IDENTIFIER ::= { atmfMediaTypes 2 }
atmfMediaSingleMode     OBJECT IDENTIFIER ::= { atmfMediaTypes 3 }
atmfMediaMultiMode      OBJECT IDENTIFIER ::= { atmfMediaTypes 4 }
atmfMediaStp            OBJECT IDENTIFIER ::= { atmfMediaTypes 5 }
atmfMediaUtp            OBJECT IDENTIFIER ::= { atmfMediaTypes 6 }

-- Traffic Descriptor Types: These types are combined with a five element
-- parameter vector to describe a Traffic Descriptor.
-- Traffic Descriptors along with a Best Effort Indicator are used to
-- indicate a Conformance Definition as defined in [TM4.0].

-- These types are no longer used
atmfNoDescriptor        OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 1 }
atmfPeakRate            OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 2 }

-- The No CLP/No SCR Type
-- Indicates the CBR.1 Conformance Definition if Best Effort is No
-- Indicates the UBR.1 and UBR.2 Conformance Definitions if Best Effort is Yes
atmfNoClpNoScr          OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 3 }
-- The use of the parameter vector for this type:
        -- Parameter #1 - peak cell rate in cells/second for CLP=0+1 traffic
```

```
        -- Parameter #2 - CDVT in tenths of microseconds
        -- Parameters #3, #4 and #5 are unused

-- These types are no longer used
atmfClpNoTaggingNoScr   OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 4 }
atmfClpTaggingNoScr     OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 5 }

-- The SCR/No CLP Type
-- Indicates the VBR.1 Conformance Definition
atmfNoClpScr            OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 6 }
-- The use of the parameter vector for this type:
        -- Parameter #1 - peak cell rate in cells/second for CLP=0+1 traffic
        -- Parameter #2 - sustainable cell rate in cells/second for CLP=0+1 traffic
        -- Parameter #3 - maximum burst size in cells
        -- Parameter #4 - CDVT in tenths of microseconds
        -- Parameter #5 - unused

-- The CLP without Tagging/SCR Type
-- Indicates the VBR.2 Conformance Definition
atmfClpNoTaggingScr     OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 7 }
-- The use of the parameter vector for this type:
        -- Parameter #1 - peak cell rate in cells/second for CLP=0+1 traffic
        -- Parameter #2 - sustainable cell rate in cells/second for CLP=0 traffic
        -- Parameter #3 - maximum burst size in cells
        -- Parameter #4 - CDVT in tenths of microseconds
        -- Parameter #5 - unused

-- The CLP with Tagging/SCR Type
-- Indicates the VBR.3 Conformance Definition
atmfClpTaggingScr       OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 8 }
-- The use of the parameter vector for this type:
        -- Parameter #1 - peak cell rate in cells/second for CLP=0+1 traffic
        -- Parameter #2 - sustainable cell rate in cells/second for CLP=0
        --      traffic, excess tagged as CLP=1
        -- Parameter #3 - maximum burst size in cells
        -- Parameter #4 - CDVT in tenths of microseconds
        -- Parameter #5 - unused

-- The ABR Type
-- Indicates the ABR Conformance Definition
atmfClpNoTaggingMcr      OBJECT IDENTIFIER ::= { atmfTrafficDescrTypes 9 }
-- The use of the parameter vector for this type:
        -- Parameter #1 - peak cell rate in cells/second
        -- parameter #2 - CDVT in tenths of microseconds
        -- Parameter #3 - minimum cell rate in cells/second
        -- Parameter #4 - unused
        -- Parameter #5 - unused

END
```

8. Link Management MIB

8.1 Overview

The Link Management MIB provides a general-purpose link management facility for ATM Interfaces. This chapter is divided into three sections: Management Information Base, Procedures, and MIB Definitions. Section 8.2 *Management Information Base* describes each of the objects in the Link Management MIB. Section 8.3 *Procedures*, describes the procedures for use of the objects described in *Management Information Base*. Finally, Section 8.4 *MIB Definitions* presents the actual MIB.

8.2 Management Information Base

The Link Management MIB provides the following groups:

- Per-System Attributes
- Per-Physical Interface Attributes
- Per-ATM Layer Interface Attributes
- Per-ATM Layer Interface Statistics
- Per-Virtual Path Attributes
- Per-Virtual Path ABR Attributes
- Per-Virtual Channel Attributes
- Per-Virtual Channel ABR Attributes
- Link Management Traps

8.2.1 Per-System Attributes (R)

No per-system object is defined in the Link Management MIB. IMEs implementing the ATM Interface MIB must support the System group defined in RFC 1213 [MIB-II], see section 6.

8.2.2 Per-Physical Interface Attributes (R)

These attributes are located in the Physical Port Group (atmfPhysicalGroup). The physical interface is identified by the interface index (atmfPortIndex).

MIB information at this level includes:

- Interface Index
- Interface Address
- Transmission Type
- Media Type
- Operational Status

- Port Specific Information
- Adjacency information

8.2.2.1 Interface Index (R)

Interface indices are used in a number of ATM Interface MIB subtrees to identify a particular physical or virtual interface on the ATM device. The Interface Index object (atmfPortIndex) has the value zero in all SNMP messages (e.g. GetRequest, GetNextRequest, GetResponse, SetRequest, and Trap) and implicitly identifies the physical or virtual interface over which ILMI messages are received. Only implicit identification is allowed. Explicit identification of interfaces using non-zero values is not allowed. Versions 3.1 and older of the User-Network Interface Specification optionally allowed one of many interfaces to be explicitly identified by an interface index unique to the ATM device. This option has been deprecated because it led to interoperability problems.

8.2.2.2 Interface Address (D)

The Interface Address object (atmfPortAddress) specified in version 2.0 of the User-Network Interface Specification is obsolete. The Address Registration extensions, specified in section 9, replace the Interface Address object. The Interface Address object should not be implemented except as required for backward compatibility.

8.2.2.3 Transmission Type (D)

The Transmission Type object (atmfPortTransmissionType) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. The Transmission Type object should not be implemented except as required for backward compatibility.

8.2.2.4 Media Type (D)

The Media Type object (atmfPortMediaType) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. The Media Type object should not be implemented except as required for backward compatibility.

8.2.2.5 Physical Layer Operational Status (D)

The Operational Status object (atmfPortOperStatus) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. The Operational Status object should not be implemented except as required for backward compatibility.

8.2.2.6 Port Specific Information (D)

The Port Specific object (atmfPortSpecific) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. The Port Specific object should not be implemented except as required for backward compatibility.

8.2.2.7 Adjacency information (R)

These objects (atmfPortMyIfName, atmfPortMyIfIdentifier, atmfMyIpNmAddress, atmfMyOsiNmNsapAddress, and atmfMySystemIdentifier) allow the neighboring system to maintain a table of adjacent systems to facilitate autodiscovery and tracing of ATM connections by Network Management Systems. It includes the system identifier that identifies the ATM device local to this IME, the interface identifier that identifies the ATM Interface managed by this IME, the address to which a Network Management Station can send Network Management protocol messages, and the name of the interface.

8.2.3 Per-ATM Layer Interface Attributes (R)

These attributes are located in the ATM Layer Group (atmfAtmLayerGroup). The interface is identified by the interface index (atmfAtmLayerIndex).

MIB information at this level includes:

- Interface Index
- Maximum Number of Active VPI Bits
- Maximum Number of Active VCI Bits
- Maximum Number of VPCs
- Maximum Number of VCCs
- Number of Configured VPCs
- Number of Configured VCCs
- Maximum SVPC VPI
- Maximum SVCC VPI
- Minimum SVCC VCI
- ATM Interface Type (Public/Private)
- ATM Device Type (User/Node)
- ILMI Version
- UNI Signalling Version
- NNI Signalling Version

8.2.3.1 Interface Index (R)

The Interface Index object (atmfAtmLayerIndex) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

8.2.3.2 Maximum Number of Active VPI Bits (R)

The Maximum Number of Active VPI Bits object (atmfAtmLayerMaxVpiBits) is the maximum number of VPI bits that may be active for this interface. Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Number of Active VPI Bits should be set to the minimum of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

If a cell is received at the local interface with inactive VPI bits set to one, the behavior of the local interface is implementation dependent.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), the Maximum Number of Active VPI bits is set to zero.

The Maximum VPI is calculated as two raised to the power of the Actual Number of Active VPI Bits, minus one.

8.2.3.3 Maximum Number of Active VCI bits (R)

The Maximum Number of Active VCI Bits object (atmfAtmLayerMaxVciBits) is the maximum number of VCI bits that may be active for this interface for the purpose of switching cells on a VCC. Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Number of Active VCI Bits should be set to the minimum of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

If a cell is received at the local interface of a VC switch with inactive VCI bits set to one, the behavior of the local interface is implementation dependent. If a cell is received at the local interface of a VP switch, this object is ignored and all VCI bits must be switched transparently.

The Maximum VCI is calculated as two raised to the power of the Actual Number of Active VCI Bits, minus one.

8.2.3.4 Maximum Number of VPCs (R)

The Maximum Number of VPCs object (atmfAtmLayerMaxVPCs) is the maximum number of switched and permanent VPCs which the local interface can support. This number includes the number of pre-configured permanent VPCs counted by atmfAtmLayerConfiguredVPCs, and MUST be at least as large as that number. Hardware and software limitations may cause this number to be less than or equal to two raised to the power of the Maximum Number of Active VPI Bits.

Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Number of VPCs should be set to the minimum of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), the Maximum Number of VPCs is set to zero.

8.2.3.5 Maximum Number of VCCs (R)

The Maximum Number of VCCs object (atmfAtmLayerMaxVCCs) is the maximum number of switched and permanent VCCs which the local interface can support. This number includes the number of pre-configured permanent VCCs counted by atmfAtmLayerConfiguredVCCs, and MUST be at least as large as that number. Hardware and software limitations may cause this number to be less than or equal to two raised to the power of the Maximum Number of Active VCI Bits plus the Maximum Number of Active VPI Bits.

Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Number of VCCs should be set to the minimum of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

8.2.3.6 Number of Configured VPCs (R)

The Number of Configured VPCs object (atmfAtmLayerConfiguredVPCs) is the current number of permanent VPCs for which the local interface is configured to process. This number represents the number of entries in the atmfVpcTable.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), the Maximum Number of VPCs is set to zero.

8.2.3.7 Number of Configured VCCs (R)

The Number of Configured VCCs object (atmfAtmLayerConfiguredVCCs) is the current number of permanent VCCs for which the local interface is configured to process. This number counts the number of entries in the atmfVccTable, which MUST include entries for all standard permanent VCCs (e.g. the Signalling, ILMI, and LECS VCCs) and non-standard permanent VCCs which are configured for use, and MUST NOT include entries for any standard permanent VCCs that are not configured for use.

> Note: Although F4 OAM flows are distinguished using unique VCIs, they are not considered as VCCs by this specification and should not be included in the atmfVccTable or counted by this number.

8.2.3.8 Maximum Switched VPC VPI (R)

The signalling stack on this ATM interface is configured to support allocation of VPIs to switched virtual path connections from a single contiguous range of VPIs beginning with VPI=1 (VPI=0 is used for ILMI and signalling VCCs), and ending with the number indicated by the Maximum Switched VPC VPI object (atmfAtmLayerMaxSvpcVpi), as shown in Figure 3.

Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Maximum Switched VPC VPI should be set to the lower of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

If switched virtual path connections are not supported on this interface, the Maximum Switched VPC VPI must be set to zero.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), the Maximum Switched VPC VPI is set to zero.

Permanent VPCs may be allocated anywhere, but it is suggested that Permanent VPCs be given VPI values greater than the Maximum SVPC VPI and less than the Maximum VPI.

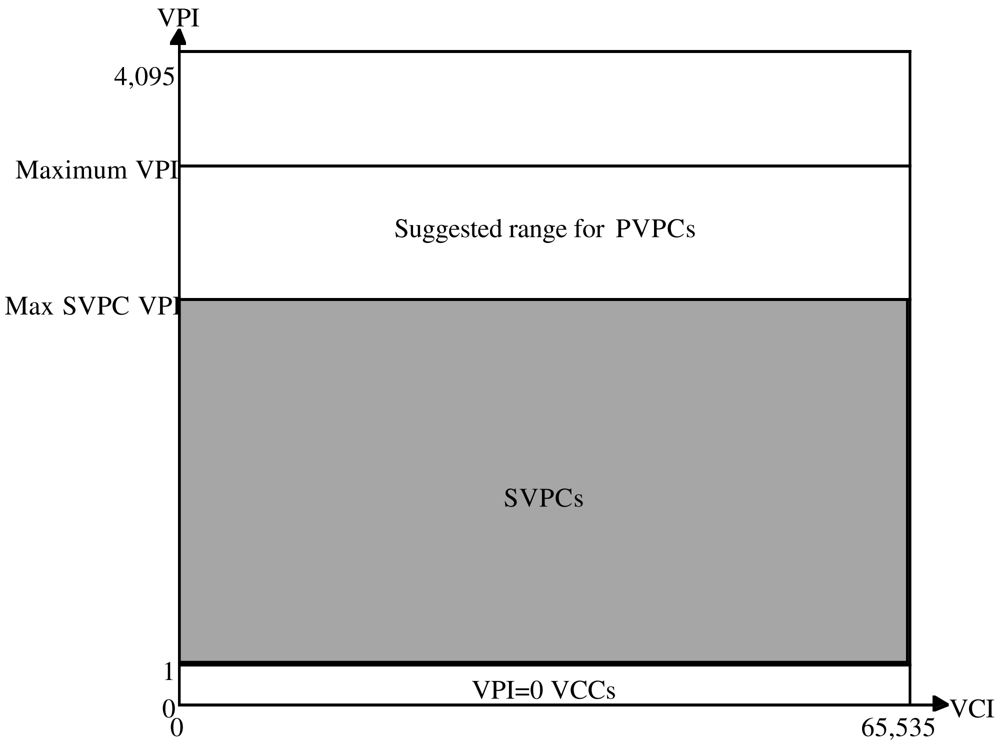


Figure 3 - Switched VPC VPI Values

8.2.3.9 Maximum Switched VCC VPI (R)

The signalling stack on this ATM interface is configured to support allocation of VPIs to switched virtual channel connections from a single contiguous range of VPIs beginning with VPI=0, and ending with the number indicated by the Maximum Switched VCC VPI object (atmfAtmLayerMaxSvccVpi). The Maximum Switched VCC VPI may be less than, equal to, or greater than the Maximum Switched VPC VPI; there is no required relationship between the values for these objects.

Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Maximum Switched VCC VPI should be set to the lower of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

If switched virtual channel connections are not supported on this interface, the Maximum Switched VCC VPI must be set to zero.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), the Maximum Switched VCC VPI is set to zero.

8.2.3.10 Minimum Switched VCC VCI (R)

The signalling stack on this ATM interface is configured to support allocation of VCIs to switched virtual channel connections from a contiguous range of VCIs beginning with the number indicated by the Minimum Switched VCC VCI object (atmfAtmLayerMinSvccVci), and ending with the Maximum VCI (see 8.2.3.3), as shown in Figure 4. The same value applies to all Switched VCC VPI values for which the signalling stack is configured (see 0).

To be compliant with ATM Forum specifications for well-known virtual circuits, this value should be at least 32.

Each IME should retrieve the value of this object from its peer at ILMI initialization, and should compare the retrieved value with the local value of this object. The Actual Minimum Switched VCC VCI should be set to the higher of the two values in order to guarantee interoperability. The local value of this object should not be modified even if the retrieved value is different, and should always remain constant unless changed through management.

If switched virtual channel connections are not supported on this interface, the Minimum Switched VCC VCI must be greater than the Maximum VCI.

Permanent VCCs may be allocated anywhere, but it is suggested that Permanent VCCs be given VCI values less than the Minimum SVCC VCI.

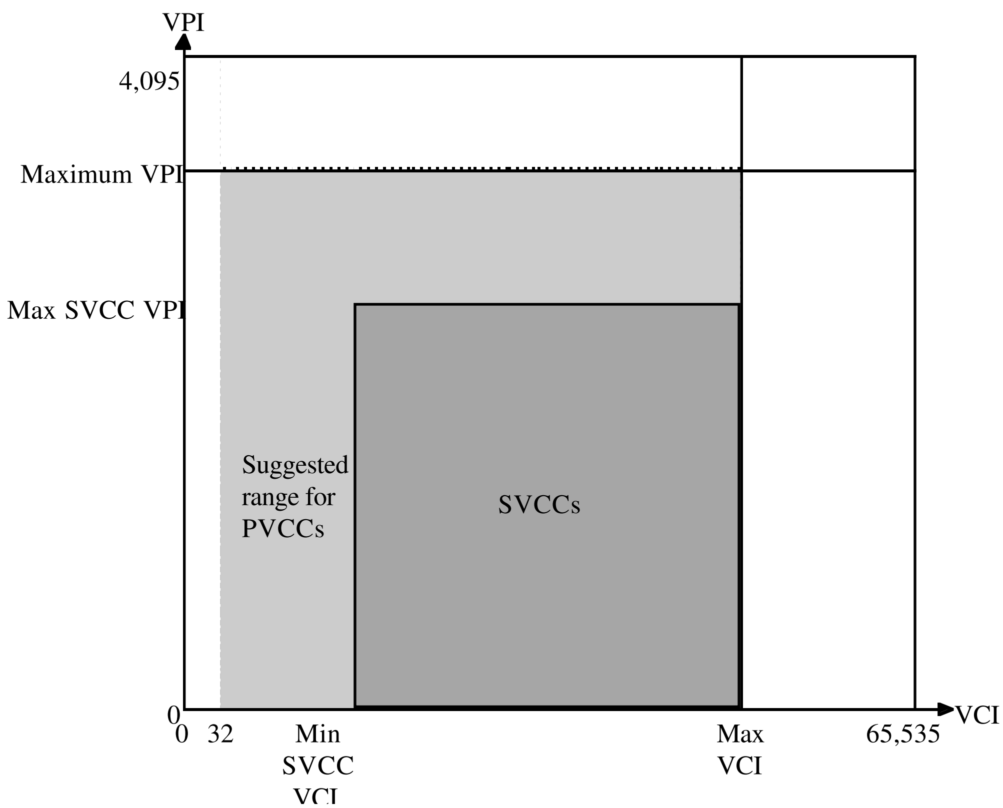


Figure 4 - Switched VCC VPI and VCI Values

8.2.3.11 ATM Public/Private Indicator (R)

The ATM Public/Private Indicator object (atmfAtmLayerUniType) indicates whether the ATM device is of the *public* or *private* type.

8.2.3.12 ATM Interface Device Type (R)

The ATM Interface Device Type object (atmfAtmLayerDeviceType) indicates whether the ATM device is of the *user* or *node* type.

8.2.3.13 ILMI Version (R)

The ILMI Version object (atmfAtmLayerIlmiVersion) indicates the latest version of the ATM Forum ILMI specification supported on this interface.

If the peer IME's value of this object is the same as, or later than the local IME's value, then the version corresponding to the local IME's value should be attempted. Otherwise, if the peer IME's value of this object is earlier, and supported locally, then the local IME should attempt the version corresponding to the peer IME's value. Otherwise, compatibility of the two IMEs cannot be assumed.

8.2.3.14 UNI Signalling Version (R)

The UNI Signalling Version object (atmfAtmLayerUniVersion) indicates the latest version of the ATM UNI Signalling specification supported on this ATM Interface.

If the peer IME’s value of this object is the same as, or later than the local IME’s value, then the version corresponding to the local IME’s value should be attempted. Otherwise, if the peer IME’s value of this object is earlier, and supported locally, then the local IME should attempt the version corresponding to the peer IME’s value. Otherwise, compatibility of the two IMEs cannot be assumed.

This object must be provided all device types including both Users and Nodes.

8.2.3.15 NNI Signalling Version (R)

The NNI Signalling Version object (atmfAtmLayerNniSigVersion) indicates the latest version of the ATM Forum PNNI signalling specification supported on this ATM Interface. Note that the PNNI routing version is not determined through ILMI.

If the peer IME’s value of this object is the same as, or later than the local IME’s value, then the version corresponding to the local IME’s value should be attempted. Otherwise, if the peer IME’s value of this object is earlier, and supported locally, then the local IME should attempt the version corresponding to the peer IME’s value. Otherwise, compatibility of the two IMEs cannot be assumed.

This object must be provided by Nodes.

8.2.4 Per-ATM Layer Interface Statistics (D)

The ATM Layer Statistics Group (atmfAtmStatsGroup) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. The ATM Layer Statistics Group should not be implemented except as required for backward compatibility.

8.2.5 Per-Virtual Path Attributes (R)

These attributes are located in the Virtual Path Group (atmfVpcGroup). This group is indexed by the interface index (atmfVpcPortIndex) and the VPI value (atmfVpcVpi), and MUST have atmfAtmLayerConfiguredVPCs entries. Only permanent virtual path connections are represented in this group. For virtual interfaces (i.e. Virtual Path Connections used by PNNI), this group is empty.

MIB information at this level includes:

- Interface Index
- VPI Value
- Operational Status
- Transmit Traffic Descriptor
- Receive Traffic Descriptor
- Best Effort Indicator
- Transmit QoS Class
- Receive QoS Class
- Service Category

8.2.5.1 Interface Index (R)

The Interface Index object (atmfVpcPortIndex) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

8.2.5.2 VPI (R)

The VPI object (atmfVpcVpi) is the value of the Virtual Path Identifier (VPI) for this VPC.

8.2.5.3 Operational Status (R)

The Operational Status object (atmfVpcOperStatus) represents the state of the VPC as known by the local device. If the end-to-end status is known then a value of end2endUp(2) or end2endDown(3) is used. If only the local status is known then a value of localUpEnd2endUnknown(4) or localDown(5) is used.

8.2.5.4 Transmit Traffic Descriptor (R)

The Transmit Traffic Descriptor is a specification of the conformance definition and associated source traffic descriptor parameter values described in [TM4.0] that are applicable to the transmit side of this interface for this VPC.

8.2.5.5 Receive Traffic Descriptor (R)

The Receive Traffic Descriptor is a specification of the conformance definition and associated source traffic descriptor parameter values described in [TM4.0] that are applicable to the receive side of this interface for this VPC.

8.2.5.6 Best Effort Indicator (R)

The Best Effort Indicator object (atmfVpcBestEffortIndicator) specifies whether best effort is requested for this VPC (see [TM4.0]).

8.2.5.7 Transmit (QoS) Class (D)

The Transmit QoS Class object (atmfVpcTransmitQoSClass) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. This object need not be implemented except as required for backward compatibility.

8.2.5.8 Receive (QoS) Class (D)

The Receive QoS Class object (atmfVpcReceiveQoSClass) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. This object need not be implemented except as required for backward compatibility.

8.2.5.9 Service Category (R)

The Service Category object (atmfVpcServiceCategory) indicates the service category of this VPC.

8.2.6 Per-Virtual Path ABR Attributes (CR)

ABR virtual path connections are "tuned" on a per-connection basis via a set of attributes. These attributes are located in the Virtual Path ABR Group (atmfVpcAbrGroup). This group is indexed by the interface index (atmfVpcAbrPortIndex) and the VPI value (atmfVpcAbrVpi). Every entry in the Virtual Path ABR Group must have a one-to-one correspondence to an entry in the Virtual Path Group.

MIB information at this level includes:

- Interface Index

- VPI Value
- ABR Operational Parameters

Support of ABR is optional. However, an interface which offers ABR service must implement all of these objects.

8.2.6.1 Interface Index (R)

The Interface Index object (atmfVpcAbrPortIndex) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

8.2.6.2 VPI (R)

The VPI object (atmfVpcAbrVpi) is the value of the Virtual Path Identifier (VPI) for this VPC.

8.2.6.3 ABR Operational Parameters (R)

The ABR VPC table (atmfVpcAbrTable) and ABR VCC table (atmfVccAbrTable) contain the ABR parameters defined in [TM4.0] which govern the ABR soruce and destination behaviors. The objects in this table are consistent with the ABR parameters defined in [TM4.0], with the following exceptions:

- Mrm and TCR are not included, as they are symbolic constants (per Table 5-1 of [TM4.0]).
- PCR and MCR are included elsewhere (in the traffic descriptor types of the object identifier definitions part of Section 7.1).
- ACR (the current rate at which a source is allowed to send cells) is not included because it is not a configured paramters and changes dynamically.
- FRTT (Fixed Round-Trip Time) and TBE (Transient Buffer Exposure) are not included because ICR (Initial Cell Rate) is specified directly by the peer IME, rather than calculated using FRTT and TBE.

The ABR operational parameters are described below.

Table 1 - ABR Operational Parameters

Name	Meaning	Brief Description
ICR	Initial Cell Rate	Upper bound on the source's transmission rate, imposed at initial start-up and after an idle period. The unit is an integer number of cells/second. The value must not exceed PCR, and is usually lower.
Nrm	Number of data cells per forward RM-cell	The maximum number of data cells a source may send between forward RM-cells. Allowed values are: 2, 4, 8, 16, 32, 64, 128, and 256.
Trm	Maximum time between forward RM-cells	Upper bound on the time between forward RM-cells for an active source (in milliseconds).
CDF	Cutoff Decrease Factor	Controls the required decrease in source transmission rate associated with lost or delayed backward RM-cells. Larger values cause a faster decrease.
RIF	Rate Increment Factor	Controls the allowed increase in source transmission rate associated with the receipt of a backward RM-cell which indicates no congestion in the network (i.e. CI=0 and NI=0). Larger values permit a faster increase.
RDF	Rate Decrease Factor	Controls the required decrease in source transmission rate associated with the receipt of a backward RM-cell indicating congestion in the network (i.e. CI=1). Larger values cause a faster decrease.
ADTF	ACR Decrease Time Factor	Allowed time between the transmission of forward RM-cells, before the source is required to decrease its transmission rate to ICR. Larger values allow a source to retain its rate longer, during periods of relative inactivity.

CRM	RM-Cells before Cutoff	Limits the number of forward RM-cells which may be sent in the absence of received backward RM-cells.

8.2.7 Per-Virtual Channel Attributes (R)

These attributes are located in the Virtual Channel Group (atmfVccGroup). This group is indexed by the interface index (atmfVccPortIndex), VCC VPI value (atmfVccVpi) and VCC VCI value (atmfVccVci), and MUST have atmfAtmLayerConfiguredVCCs entries. Only permanent virtual channel connections are represented in this group, including all standard permanent VCCs (e.g. the Signalling, ILMI, and LECS VCCs) and non-standard permanent VCCs which are configured for use, but not including any standard permanent VCCs that are not configured for use.

> Note: Although F4 OAM flows are distinguished using unique VCIs, they are not considered as VCCs by this specification and should not be included in this group.

MIB information at this level includes:

- Interface Index
- VPI/VCI Value
- Operational Status
- Transmit Traffic Descriptor
- Receive Traffic Descriptor
- Best Effort
- Transmit QoS Class
- Receive QoS Class
- Transmit Frame Discard Indication
- Receive Frame Discard Indication
- Service Category

8.2.7.1 Interface Index (R)

The Interface Index object (atmfVccPortIndex) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

8.2.7.2 VPI (R)

The VPI object (atmfVccVpi) is the value of the Virtual Path Identifier (VPI) for this VCC.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), this object has the value zero in all SNMP messages (e.g. GetRequest, GetNextRequest, GetResponse, SetRequest, and Trap) and implicitly identifies the VPI over which ILMI messages are received.

8.2.7.3 VCI (R)

The VCI object (atmfVccVci) is the value of the Virtual Channel Identifier (VCI) for this VCC.

8.2.7.4 Operational Status (R)

The Operational Status object (atmfVccOperStatus) represents the state of the VCC as known by the local device. If the end-to-end status is known then a value of end2endUp(2) or end2endDown(3) is used. If only the local status is known then a value of localUpEnd2endUnknown(4) or localDown(5) is used.

8.2.7.5 Transmit Traffic Descriptor (R)

This is a specification of the conformance definition and associated source traffic descriptor parameter values described in [TM4.0] that are applicable to the transmit side of this interface for this VCC.

8.2.7.6 Receive Traffic Descriptor (R)

This is a specification of the conformance definition and associated source traffic descriptor parameter values described in [TM4.0] that are applicable to the receive side of this interface for this VCC.

8.2.7.7 Best Effort Indicator (R)

The Best Effort Indicator object (atmfVccBestEffortIndicator) specifies whether best effort is requested for this VCC (see [TM4.0]).

8.2.7.8 Transmit (QoS) Class (D)

The Transmit QoS Class object (atmfVccTransmitQoSClass) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. This object need not be implemented except as required for backward compatibility.

8.2.7.9 Receive (QoS) Class (D)

The Receive QoS Class object (atmfVccReceiveQoSClass) specified in versions 3.1 and older of the User-Network Interface Specification is deprecated. This object need not be implemented except as required for backward compatibility.

8.2.7.10 Transmit Frame Discard Indication (R)

The Transmit Frame Discard Indication object (atmfVccTransmitFrameDiscard) specifies whether the network is allowed to invoke frame discard mechanisms in the transmit direction of the associated connection, as specified in [TM4.0].

8.2.7.11 Receive Frame Discard Indication (R)

The Receive Frame Discard Indication object (atmfVccReceiveFrameDiscard) specifies whether the network is allowed to invoke frame discard mechanisms in the receive direction of the associated connection, as specified in [TM4.0].

8.2.7.12 Service Category (R)

The Service Category object (atmfVccServiceCategory) indicates the service category of this VCC.

8.2.8 Per-Virtual Channel ABR Attributes (CR)

The Virtual Channel ABR source and destination behaviors are "tuned" on a per-connection basis via a set of attributes. These attributes are located in the Virtual Channel ABR Group (atmfVccAbrGroup). This group is indexed by the interface index (atmfVccAbrPortIndex), VCC VPI value (atmfVccAbrVpi) and VCC VCI value (atmfVccAbrVci). Every entry in the Virtual Channel ABR Group must have a one-to-one correspondence to an entry in the Virtual Channel Group.

MIB information at this level includes:

- Interface Index
- VPI/VCI Value
- ABR Operational Parameters

Support of ABR is optional. However, an interface which offers ABR service must implement all of these objects.

8.2.8.1 Interface Index (R)

The Interface Index object (atmfVccAbrPortIndex) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

8.2.8.2 VPI (R)

The VPI object (atmfVccAbrVpi) is the value of the Virtual Path Identifier (VPI) for this VPC.

For virtual interfaces (i.e. Virtual Path Connections used by PNNI), this object has the value zero in all SNMP messages (e.g. GetRequest, GetNextRequest, GetResponse, SetRequest, and Trap) and implicitly identifies the VPI over which ILMI messages are received.

8.2.8.3 VCI (R)

The VCI object (atmfVccAbrVci) is the value of the Virtual Channel Identifier (VCI) for this VCC.

8.2.8.4 ABR Operational Parameters (R)

The ABR operational parameters used with VCCs, are identical to those used with VPCs. See section 8.2.6.3 for a description.

8.2.9 Link Management Traps (R)

Two traps (atmfVpcChange and atmfVccChange) have been defined for the ILMI in order to indicate a newly configured, modified, or deleted permanent VPC or permanent VCC. The atmfVpcChange trap provides the Virtual Path Identifier (VPI) value of the new or deleted configured VPC at an ATM Interface. The atmfVccChange trap provides the Virtual Channel Identifier (VCI) and the VPI values of the new or deleted configured VCC at an ATM Interface.

8.3 Procedures

8.3.1 ILMI Connectivity Procedures (CR)

ILMI Connectivity Procedures are used to detect the establishment and subsequent loss of ILMI Connectivity. These events are used by the auto-configuration (section 8.3.3) and address registration procedures (section 9.5). Establishment or loss of ILMI Connectivity does not have any effect on PVCs, UNI or PNNI Signalling, or on the UNI or PNNI Signalling AAL [UNI4.0, PNNI1.0], except as specified in section 8.3.3. ILMI Connectivity Procedures are used with both physical links and virtual links (i.e. Virtual Path Connections used by PNNI). ILMI Connectivity Procedures are required for an ATM node, and are optional for an ATM user.

Unless administratively disabled, ILMI Connectivity MUST periodically be tested while Link Connectivity is established. Loss of ILMI connectivity MUST be assumed whenever a link failure is indicated (e.g. via physical link carrier loss, VP OAM procedures, etc.). Normal hysterisis mechanisms may be implemented to prevent rapid cycling of Link Connectivity.

For the purposes of establishing, verifying, or re-establishing ILMI Connectivity, the IME shall transmit a single GetRequest or GetNextRequest for the objects atmfPortMyIfIdentifier, atmfMySystemIdentifier, and sysUpTime. A matching ILMI response message indicates that ILMI Connectivity is (still) established. The returned values of the requested objects are used by the Change of Attachment Point Detection Procedures, as described in section 8.3.2.

In order to detect the establishment of ILMI Connectivity, the IME shall test connectivity with its peer IME every S seconds, until an ILMI response message is received indicating that ILMI Connectivity is established. This will continue while Link Connectivity is maintained. Once an ILMI response message is received, auto-configuration should begin, as described in section 8.3.4. On completion of auto-configuration, address registration may take place, as necessary, as described in section 9.5.

To detect subsequent loss of ILMI Connectivity, an IME shall test connectivity with its peer IME every T seconds. ILMI Connectivity is declared lost when no ILMI response message is received for K consecutive polls.

To regain ILMI connectivity, the IME shall test connectivity with its peer IME every S seconds, or until a link failure occurs. The IME shall indicate that ILMI Connectivity is re-established after receiving an ILMI response message, and shall restart auto-configuration and address registration.

The default values are K=4, S=1, and T=5; different values can be configured.

The IME shall not declare ILMI Connectivity to be lost due to any change in the status of the Signalling AAL.

8.3.2 Change of Attachment Point Detection Procedures (O)

Change of Attachment Point Detection Procedures are used to detect a change of attachment point. Signalling may withstand physical layer failures for a short period of time before releasing connections. During this brief time, a small chance exists for two links to be swapped without detection by signalling. ILMI's Change of Attachment Point Detection Procedures can reliably detect this event. Unlike an establishment or loss of ILMI Connectivity, detection of a change of attachment point *does* have an effect on UNI and PNNI Signalling, and MUST result in the immediate release of all SVCs.

If Change of Attachment Point Detection Procedures are implemented, an IME MUST keep local copies of its peer's atmfPortMyIfIdentifier atmfMySystemIdentifier, and sysUpTime objects, and MUST initialize these variables to an illegal value upon the initial establishment of ILMI Connectivity. An IME MUST NOT re-initialize these variables upon the loss or subsequent re-establishment of ILMI Connectivity.

As described in section 8.3.1, an IME reads the current values of the atmfPortMyIfIdentifier atmfMySystemIdentifier, and sysUpTime objects from its peer IME in order to establish, verify, or re-establish ILMI Connectivity. An IME should compare the current values of these objects, returned in an ILMI response message, with the local copies of these objects. If the returned values of atmfPortMyIfIdentifier or atmfMySystemIdentifier are different from the local copies, or if the returned value of sysUpTime is less than the local copy, the IME shall declare that the attachment point has changed and shall:

1. Set the local copies of its peer's atmfPortIdentifier atmfMySystemIdentifier, and sysUpTime objects to the returned values.
2. Declare ILMI Connectivity to be lost.
3. Inform its corresponding UNI [UNI4.0] or PNNI [PNNI1.0] signalling entity to release all SVCs controlled by that entity. The method used to inform the signalling entity is implementation dependent.
4. Send a coldStart Trap to its peer IME entity to indicate the interface is restarting.
5. Declare ILMI Connectivity to be re-established.
6. Reinvoke any automatic configuration procedures.
7. Perform address registration, if necessary.

A pre-ILMI 4.0 IME may return a noSuchName error to the request for the atmfPortIndex and atmfMySystemIdentifier objects. A noSuchName error indicates that ILMI Connectivity is (still) established. ILMI's Change of Attachment Point Detection Procedures should consider a noSuchName error as a special, unique value for these objects. It must not match the illegal value that the local copies of these variables were initialized to, nor should it match any previous successfully returned values, but it should match any previously returned noSuchName error.

If the attachment point has not changed, the IME shall set the local copy of its peer's sysUpTime objects to the returned value. The sysUpTime should be a constantly increasing value in the normal case.

All IMEs MAY perform this procedure.

8.3.3 Secure Link Procedures (O)

As described above, signalling may not drop some of its connections when an attachment point changes. If the new attachment point is not running ILMI (e.g. an IISP 1.0 link), the local IME will detect a loss of ILMI Connectivity, rather than a change of attachment point. This may be unacceptable for secure links. To guard against this breach of security, an IME for a secure link MAY decide to process a loss of ILMI Connectivity as a change of attachment point. When a loss of ILMI Connectivity is detected, the IME should inform its corresponding UNI [UNI4.0] or

PNNI [PNNI1.0] signalling entity to release all SVCs controlled by that entity, in addition to executing the normal loss of ILMI Connectivity procedures. The method used to inform the signalling entity is implementation dependent.

8.3.4 Automatic Configuration Procedures (O)

8.3.4.1 ATM Interface Type and IME Type

A private user or node MAY automatically configure the type of its ATM Interface(s) as a Public UNI, Private UNI, or PNNI, and the type of IME as User-Side, Network-Side, or Symmetric. Automatic configuration procedures for a public node are for further study. If automatic configuration is used, the user or node performing the procedure should not send any non-ILMI messages over the ATM Interface until the automatic configuration procedure has completed. This interface-type information may be used by other ATM protocols, such as ATM Forum's UNI signalling [UNI4.0] and PNNI signalling and routing protocols [PNNI1.0] to assist those protocols' operation.

As part of the auto-configuration procedures, if the atmfAtmLayerUniType object for a port on an ATM device has the value private, then the IME shall retrieve the values of atmfAtmLayerDeviceType and atmfAtmLayerUniType from the peer IME and determine the value of the Interface Type according to Table 2.

Table 2 - Interface-Type Determination for Private Devices[1]

Local DeviceType[2]	Peer DeviceType[3]	Peer UniType[4]	Interface Type[5]	IME Type[6]
user(1)	noSuchName	public(1)	Public UNI	User-Side
user(1)	noSuchName	private(2)	Private UNI	User-Side
user(1)	node(2)	public(1)	Public UNI	User-Side
user(1)	node(2)	private(2)	Private UNI	User-Side
user(1)	user(1)	N/A	Undefined[7]	Undefined[7]
node(2)	noSuchName	public(1)	Public UNI	User-Side
node(2)	noSuchName	private(2)	Private UNI	Network-Side
node(2)	node(2)	public(1)[9]	Public UNI	User-Side
node(2)	node(2)	private(2)[9]	See Note 10	See Note 11
node(2)	user(1)	public(1)	Undefined[8]	Undefined[8]
node(2)	user(1)	private(2)	Private UNI	Network-Side

Note 1 - Interface Type determination for public devices is outside the scope of this document.
Note 2 - Value retrieved from local atmfAtmLayerUniType and atmfAtmLayerDeviceType objects.
Note 3 - Value retrieved from remote atmfAtmLayerDeviceType object.
Note 4 - Value retrieved from remote atmfAtmLayerUniType object.
Note 5 - Resulting Interface Type.
Note 6 - Resulting IME Type.
Note 7 - The result of connecting a user to a user is undefined.
Note 8 - The result of connecting a Private Node to a Public User is undefined.
Note 9 - A Public Network ATM switch desiring to support a PNNI Interface Type must indicate its atmfAtmLayerUniType as private(2).
Note 10 - The resulting Interface Type is IISP if the lesser of the local and peer atmfAtmLayerNniSigVersion is iisp(2), or PNNI if the lesser of the local and peer atmfAtmLayerNniSigVersion is pnniVersion1point0(3). In the case of IISP, the UNI signalling version upon which IISP is based is determined by the lesser of the local and peer atmfAtmLayerUniVersion.
Note 11 - The resulting IME Type is User-Side if the Interface Type is IISP and the local atmfMySystemIdentifier is larger than the peer IME's atmfMySystemIdentifier, Network-Side if the Interface Type is IISP and the local atmfMySystemIdentifier is smaller than the peer IME's atmfMySystemIdentifier, and Symmetric if Interface Type is PNNI.

8.3.4.2 Per-ATM Layer Interface Attributes

An IME may also automatically configure a number of other Per-ATM Layer Interface Attributes. Procedures for auto-configuration of each object are specified in section 8.2.3.

8.3.4.3 Signalling VCC Transmission Parameters

[UNI4.0] defines default traffic parameter values for the signalling VCC. This section specifies procedures which may be used to automatically configure different values for use on a particular interface. Implementation of this procedure is optional.

Prior to attempting to establish ILMI Connectivity, the IME shall initialize its local values of atmfVccServiceCategory, atmfVccBestEffortIndicator, atmfVccReceiveTrafficDescriptorType, atmfVccReceiveTrafficDescriptorParam1, atmfVccReceiveTrafficDescriptorParam2, atmfVccReceiveTrafficDescriptorParam3, atmfVccReceiveTrafficDescriptorParam4, and atmfVccReceiveTrafficDescriptorParam5 objects for VPI=0, VCI=5. These values should not not be modified as a result of the procedures described in this section. When local modifications are made to these parameters for other reasons, e.g. through management, an appropriate Link Management Trap must be issued to notify the peer IME.

Before the signalling channel is initialized, the IME shall read the value of the peer IME's atmfVccServiceCategory, atmfVccBestEffortIndicator, atmfVccReceiveTrafficDescriptorType, atmfVccReceiveTrafficDescriptorParam1, atmfVccReceiveTrafficDescriptorParam2, atmfVccReceiveTrafficDescriptorParam3, atmfVccReceiveTrafficDescriptorParam4, and atmfVccReceiveTrafficDescriptorParam5 objects for VPI=0, VCI=5.

If the atmfVccServiceCategory object does not exist, or the value read is invalid, but a valid value is read from the atmfVccReceiveTrafficDescriptorType object, the IME shall assume a value for the atmfVccServiceCategory as shown in Table 3:

Table 3 - Default atmfVccServiceCategory

atmfVccReceiveTrafficDescriptorType	Default atmfVccServiceCategory
atmfNoClpNoScr	ubr(6)
atmfNoClpScr	nrtVbr(4)
atmfClpNoTaggingScr	nrtVbr(4)
atmfClpTaggingScr	nrtVbr(4)
atmfClpNoTaggingMcr	abr(5)

After determining the Service Category as described above, the node shall determine if one of the valid combinations shown in Table 4 has been read.

Table 4 - Valid Transmission Parameter Combinations

atmfVccServiceCategory	atmfVccReceiveTrafficDescriptorType	atmfVccBestEffortIndicator
cbr(2)	atmfNoClpNoScr	false(2)
rtVbr(3)	atmfNoClpScr or atmfClpNoTaggingScr or atmfClpTaggingScr	false(2)
nrtVbr(4)	atmfNoClpScr or atmfClpNoTaggingScr or atmfClpTaggingScr	false(2)
abr(5)	atmfClpNoTaggingMcr	false(2)
ubr(6)	atmfNoClpNoScr	true(1)

If a valid combination has been read and the service categories and conformance definitions are the same in both IMEs for a particular direction, then the IME shall determine the traffic contract for the signalling VCC by taking for each traffic parameter the minimum of its local atmfVccTransmitTrafficDescriptorParam value and the peer’s atmfVccReceiveTrafficDescriptorParam value, except for the CDVT which is determined only by the peer's atmfVccReceiveTrafficDescriptorParam value. The resulting traffic contract is used for resource allocation as well as for traffic shaping applied to ensure conformance.

If a valid combination can not be determined, the behavior is undefined.

> Note: If an IME reads an atmfVccServiceCategory value from its peer which is different from the value with which it is itself configured, then the behavior is undefined. However a node which has its signalling channel receive parameters configured nrtVbr or rtVbr should in most cases be able to operate signalling successfully with a node which has its transmitter configured UBR, provided that it does not police its received cell stream. Similarly, a node configured nrtVbr will in almost all cases work successfully with a node configured rtVbr.
>
> The behavior is also undefined if there is a mismatch in the conformance definitions implied by the two configurations.

8.3.5 Modification of Local Attributes (R)

When local modifications are made to the Virtual Path Group or the Virtual Channel Group, e.g. through management, an appropriate Link Management Trap is issued to notify the peer IME. When the value of other objects, e.g. per-ATM Layer Interface Attributes, is modified, the IME MUST be re-initialized in order to effect the change. The IME shall:

1. Declare ILMI Connectivity to be lost.
2. Inform its UNI [UNI4.0] or PNNI [PNNI1.0] signalling entity to disconnect all connections.
3. Reset the local sysUpTime to zero.
4. Send a coldStart Trap to its peer IME entity to indicate the interface is restarting.
5. Reinvoke any automatic configuration procedure.
6. Perform address registration, if necessary.
7. Declare ILMI Connectivity to be re-established.

8.4 MIB Definitions

```
ATM-FORUM-MIB DEFINITIONS ::= BEGIN

IMPORTS
        atmForum,
        TruthValue,
        ClnpAddress,
        AtmServiceCategory,
        atmfPhysicalGroup,
        atmfAtmLayerGroup,
        atmfAtmStatsGroup,
        atmfVpcGroup,
        atmfVccGroup,
        atmfVpcAbrGroup,
        atmfVccAbrGroup                  FROM ATM-FORUM-TC-MIB
        Counter, IpAddress               FROM RFC1155-SMI
        DisplayString, PhysAddress       FROM RFC1213-MIB
        TRAP-TYPE                        FROM RFC-1215
        OBJECT-TYPE                      FROM RFC-1212;


--              Textual Conventions

-- All representations of ATM addresses in this MIB Module use
-- the data type:
AtmAddress ::= OCTET STRING (SIZE (0 .. 32))
-- Note this data type is used only by the deprecated object
-- atmfPortAddress. Another definition (a refined one) is
```

```
-- specified in the Textual Conventions MIB.


--              The Physical Port Group
--
-- This group is mandatory for all ATM Interface devices.
--
-- The Physical Port Table

atmfPortTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfPortEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of physical layer status and parameter
                information for the ATM Interface’s physical interface."
        ::= { atmfPhysicalGroup 1 }

atmfPortEntry OBJECT-TYPE
        SYNTAX  AtmfPortEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing information about
                the physical layer of an ATM Interface."
        INDEX   { atmfPortIndex }
        ::= { atmfPortTable 1 }

AtmfPortEntry ::=
        SEQUENCE {
                atmfPortIndex
                        INTEGER,
                atmfPortAddress
                        AtmAddress,
                atmfPortTransmissionType
                        OBJECT IDENTIFIER,
                atmfPortMediaType
                        OBJECT IDENTIFIER,
                atmfPortOperStatus
                        INTEGER,
                atmfPortSpecific
                        OBJECT IDENTIFIER,
                atmfPortMyIfName
                        DisplayString,
                atmfPortMyIfIdentifier
                        INTEGER

        }

atmfPortIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of
                identifying the ATM Interface over which this message
                was received."
        ::= { atmfPortEntry 1 }

atmfPortAddress OBJECT-TYPE
        SYNTAX  AtmAddress
        ACCESS  read-only
        STATUS  obsolete
        DESCRIPTION
```

```
                "This object should not be implemented except as
                required for backward compatibility with version 2.0
                of the UNI specification. The Address Group, defined
                in theAddress Registration MIB should be used instead."
        ::= { atmfPortEntry 2 }

atmfPortTransmissionType OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification. Appropriate Network Management
                MIBs should be used instead."
        ::= { atmfPortEntry 3 }

atmfPortMediaType OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification. Appropriate Network Management
                MIBs should be used instead."
        ::= { atmfPortEntry 4 }

atmfPortOperStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                        other(1),
                        inService(2),
                        outOfService(3),
                        loopBack(4)
                }
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification. Appropriate Network Management
                MIBs should be used instead."
        ::= { atmfPortEntry 5 }

atmfPortSpecific OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification. Appropriate Network Management
                MIBs should be used instead."
        ::= { atmfPortEntry 6 }

atmfPortMyIfName OBJECT-TYPE
        SYNTAX  DisplayString
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "A textual name of this interface. If this system is
                manageable through SNMP, and supports the object
                ifName, the value of this object must be identical
                with that of ifName for the ifEntry of the lowest
```

```
                level physical interface for this port. This interface
                must be uniquely named on this system to distinguish
                parallel links with a neighboring system.  If this
                interface does not have a textual name, the value of
                this object is a zero length string."
        ::= { atmfPortEntry 7 }

atmfPortMyIfIdentifier OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "A unique value for each ATM interface. The scheme used to
                number the ATM interfaces on an ATM device is
                implementation specific. One way to generate this value is
                to use the ifIndex that an SNMP manager would use to
                identify the port. This interface must be uniquely numbered
                on this system to distinguish parallel links with a
                neighboring system."
        ::= { atmfPortEntry 8 }

--  Note: The typical IME will support only one of the following two objects
atmfMyIpNmAddress OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An IP Address to which a Network Management Station
                can send Network Management protocol messages, e.g. SNMP
                messages to UDP port 161, in order to access network
                management information concerning the operation of the
                ATM device local to this IME. If this object is supported,
                but the Network Management Agent has not been configured with
                an IP Address, the IME should return 0.0.0.0."
        ::= { atmfPhysicalGroup 2 }

atmfMyOsiNmNsapAddress OBJECT-TYPE
        SYNTAX  ClnpAddress
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An NSAP Address to which a Network Management Station
                can send Network Management protocol messages in order
                to access network management information concerning
                the operation of the ATM device local to this IME.
                If this object is supported,
                but the Network Management Agent has not been configured with
                an NSAP Address, the IME should return 0.0.0.0"
        ::= { atmfPhysicalGroup 3 }

atmfMySystemIdentifier OBJECT-TYPE
        SYNTAX  OCTET STRING (SIZE (6))
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "A 48 bit identifier, taken from the IEEE universally
                administered MAC address space, which uniquely
                identifies the ATM device local to this IME."
        ::= { atmfPhysicalGroup 4 }


--              The ATM Layer Group
-- This group is mandatory for all ATM Interfaces.
--
```

```
-- ATM-layer specific information for the ATM Interface.

atmfAtmLayerTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfAtmLayerEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of ATM layer status and parameter information
                for the ATM Interface."
        ::= { atmfAtmLayerGroup 1 }

atmfAtmLayerEntry OBJECT-TYPE
        SYNTAX  AtmfAtmLayerEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing information about
                the ATM layer of an ATM Interface."
        INDEX   { atmfAtmLayerIndex }
        ::= { atmfAtmLayerTable 1 }

AtmfAtmLayerEntry ::=
        SEQUENCE {
                atmfAtmLayerIndex
                        INTEGER,
                atmfAtmLayerMaxVPCs
                        INTEGER,
                atmfAtmLayerMaxVCCs
                        INTEGER,
                atmfAtmLayerConfiguredVPCs
                        INTEGER,
                atmfAtmLayerConfiguredVCCs
                        INTEGER,
                atmfAtmLayerMaxVpiBits
                        INTEGER,
                atmfAtmLayerMaxVciBits
                        INTEGER,
                atmfAtmLayerUniType
                        INTEGER,
                atmfAtmLayerUniVersion
                        INTEGER,
                atmfAtmLayerDeviceType
                        INTEGER,
                atmfAtmLayerIlmiVersion
                        INTEGER,
                atmfAtmLayerNniSigVersion
                        INTEGER,
                atmfAtmLayerMaxSvpcVpi
                        INTEGER,
                atmfAtmLayerMaxSvccVpi
                        INTEGER,
                atmfAtmLayerMinSvccVci
                        INTEGER
        }

atmfAtmLayerIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of
                identifying the ATM Interface over which this message
                was received."
        ::= { atmfAtmLayerEntry 1 }
```

```
atmfAtmLayerMaxVPCs OBJECT-TYPE
        SYNTAX  INTEGER (0..4096)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of switched and permanent VPCs
                supported on this ATM Interface. For virtual interfaces
                (i.e. Virtual Path Connections), the maximum number of VPCs
                PNNI may communicate over is set to zero."
        ::= { atmfAtmLayerEntry 2 }

atmfAtmLayerMaxVCCs OBJECT-TYPE
        SYNTAX  INTEGER (0..268435456)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of switched and permanent VCCs
                supported on this ATM Interface."
        ::= { atmfAtmLayerEntry 3 }

atmfAtmLayerConfiguredVPCs OBJECT-TYPE
        SYNTAX  INTEGER (0..4096)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of permanent VPCs configured for use on
                this ATM Interface. For virtual interfaces (i.e. Virtual Path
                Connections used by PNNI), the maximum number of VPCs
                is set to zero."
        ::= { atmfAtmLayerEntry 4 }

atmfAtmLayerConfiguredVCCs OBJECT-TYPE
        SYNTAX  INTEGER (0.. 268435456)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of permanent VCCs configured for use on
                this ATM Interface."
        ::= { atmfAtmLayerEntry 5 }

atmfAtmLayerMaxVpiBits OBJECT-TYPE
        SYNTAX  INTEGER (0..12)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of active VPI bits on this ATM Interface.
                For virtual interfaces (i.e. Virtual Path Connections used by PNNI),
                this value has no meaning and is set to zero."
        ::= { atmfAtmLayerEntry 6 }

atmfAtmLayerMaxVciBits OBJECT-TYPE
        SYNTAX  INTEGER (0..16)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of active VCI bits on this ATM Interface."
        ::= { atmfAtmLayerEntry 7 }

atmfAtmLayerUniType OBJECT-TYPE
        SYNTAX  INTEGER { public(1), private(2) }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
```

```
                "The type of the ATM device, either public or private."
        ::= { atmfAtmLayerEntry 8 }

atmfAtmLayerUniVersion OBJECT-TYPE
        SYNTAX  INTEGER {
                        version2point0(1),
                        version3point0(2),
                        version3point1(3),
                        version4point0(4),
                        unsupported(5)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An indication of the latest version of the ATM Forum UNI
                Signalling Specification that is supported on this ATM
                Interface.  If this value is not present, a version of the UNI
                earlier than 3.1 is supported.

                If the peer IME’s value of this object is the same as,
                or later than the local IME’s value, then the version
                corresponding to the local IME’s value should be
                attempted.  Otherwise, if the peer IME’s value of this
                object is earlier, and supported locally, then the
                local IME should attempt the version corresponding to
                the peer IME’s value.  Otherwise, compatibility of the
                two IMEs cannot be assumed."
        ::= { atmfAtmLayerEntry 9 }

atmfAtmLayerDeviceType OBJECT-TYPE
        SYNTAX  INTEGER { user(1), node(2) }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "This object determines the type of the ATM device. This
                object is used in automatic ATM Interface-Type determination
                procedure such that a correct operational ATM
                Interface-type can be determined. An ATM End System
                shall take the value of user(1), and an ATM network
                node shall take the value of node(2)."
        ::= { atmfAtmLayerEntry 10 }

atmfAtmLayerIlmiVersion OBJECT-TYPE
        SYNTAX  INTEGER { unsupported(1), version4point0(2) }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An indication of the latest version of the ATM Forum
                ILMI Specification that is supported on this ATM Interface.

                If the peer IME’s value of this object is the same as,
                or later than the local IME’s value, then the version
                corresponding to the local IME’s value should be
                attempted.  Otherwise, if the peer IME’s value of this
                object is earlier, and supported locally, then the
                local IME should attempt the version corresponding to
                the peer IME’s value.  Otherwise, compatibility of the
                two IMEs cannot be assumed.

                If this object is not present, a version of the ILMI earlier
                than 4.0 is supported."
        ::= { atmfAtmLayerEntry 11 }

atmfAtmLayerNniSigVersion OBJECT-TYPE
```

```
        SYNTAX  INTEGER { unsupported(1), iisp(2),
                          pnniVersion1point0(3) }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An indication of the latest version of the ATM Forum
                PNNI Signalling Specification that is supported on this
                ATM Interface. Note that the PNNI routing version is not
                determined through ILMI.

                If the peer IME's value of this object is the same as,
                or later than the local IME's value, then the version
                corresponding to the local IME's value should be
                attempted.  Otherwise, if the peer IME's value of this
                object is earlier, and supported locally, then the
                local IME should attempt the version corresponding to
                the peer IME's value.  Otherwise, compatibility of the
                two IMEs cannot be assumed."
        ::= { atmfAtmLayerEntry 12 }

atmfAtmLayerMaxSvpcVpi OBJECT-TYPE
        SYNTAX  INTEGER (0..4096)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
               "The maximum VPI that the signalling stack on the ATM
                interface is configured to support for allocation to
                switched virtual path connections."
        ::= { atmfAtmLayerEntry 13 }

atmfAtmLayerMaxSvccVpi OBJECT-TYPE
        SYNTAX  INTEGER (0..4096)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum VPI that the signalling stack on the ATM
                interface is configured to support for allocation to
                switched virtual channel connections."
        ::= { atmfAtmLayerEntry 14 }

atmfAtmLayerMinSvccVci OBJECT-TYPE
        SYNTAX  INTEGER (0..65536)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "This is the minimum VCI value that the
                signalling stack is configured to support for
                allocation to switched virtual channel connections.  The same
                value applies to all SVCC VPI values for which the
                signalling stack is configured."
        ::= { atmfAtmLayerEntry 15 }


--              The ATM Statistics Group
-- This group is deprecated and should not be implemented except as
-- required for backward compatibility with version 3.1 of the UNI
-- specification.

atmfAtmStatsTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfAtmStatsEntry
        ACCESS  not-accessible
        STATUS  deprecated
        DESCRIPTION
                "This group is deprecated and should not be implemented
```

```
                 except as required for backward compatibility with version
                 3.1 of the UNI specification."
        ::= { atmfAtmStatsGroup 1 }

atmfAtmStatsEntry OBJECT-TYPE
        SYNTAX  AtmfAtmStatsEntry
        ACCESS  not-accessible
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        INDEX   { atmfAtmStatsIndex }
        ::= { atmfAtmStatsTable 1 }

AtmfAtmStatsEntry ::=
        SEQUENCE {
                atmfAtmStatsIndex
                        INTEGER,
                atmfAtmStatsReceivedCells
                        Counter,
                atmfAtmStatsDroppedReceivedCells
                        Counter,
                atmfAtmStatsTransmittedCells
                        Counter
        }

atmfAtmStatsIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfAtmStatsEntry 1 }

atmfAtmStatsReceivedCells OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfAtmStatsEntry 2 }

atmfAtmStatsDroppedReceivedCells OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfAtmStatsEntry 3 }

atmfAtmStatsTransmittedCells OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
```

```
                of the UNI specification."
        ::= { atmfAtmStatsEntry 4 }


--              The Virtual Path Group
-- This group is mandatory for all ATM Interfaces.
--
-- Information concerning Virtual Path Connections

atmfVpcTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfVpcEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of status and parameter information on the
                virtual path connections which cross this ATM
                Interface. There is one entry in this table for each
                permanent virtual path connection."
        ::= { atmfVpcGroup 1 }

atmfVpcEntry OBJECT-TYPE
        SYNTAX  AtmfVpcEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing information about a
                particular virtual path connection."
        INDEX   { atmfVpcPortIndex, atmfVpcVpi }
        ::= { atmfVpcTable 1 }

AtmfVpcEntry ::=
        SEQUENCE {
                atmfVpcPortIndex
                        INTEGER,
                atmfVpcVpi
                        INTEGER,
                atmfVpcOperStatus
                        INTEGER,
                atmfVpcTransmitTrafficDescriptorType
                        OBJECT IDENTIFIER,
                atmfVpcTransmitTrafficDescriptorParam1
                        INTEGER,
                atmfVpcTransmitTrafficDescriptorParam2
                        INTEGER,
                atmfVpcTransmitTrafficDescriptorParam3
                        INTEGER,
                atmfVpcTransmitTrafficDescriptorParam4
                        INTEGER,
                atmfVpcTransmitTrafficDescriptorParam5
                        INTEGER,
                atmfVpcReceiveTrafficDescriptorType
                        OBJECT IDENTIFIER,
                atmfVpcReceiveTrafficDescriptorParam1
                        INTEGER,
                atmfVpcReceiveTrafficDescriptorParam2
                        INTEGER,
                atmfVpcReceiveTrafficDescriptorParam3
                        INTEGER,
                atmfVpcReceiveTrafficDescriptorParam4
                        INTEGER,
                atmfVpcReceiveTrafficDescriptorParam5
                        INTEGER,
                atmfVpcQoSCategory
                        INTEGER,
```

```
                atmfVpcTransmitQoSClass
                        INTEGER,
                atmfVpcReceiveQoSClass
                        INTEGER,
                atmfVpcBestEffortIndicator
                        TruthValue,
                atmfVpcServiceCategory
                        AtmServiceCategory
        }

atmfVpcPortIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of
                identifying the ATM Interface over which this message
                was received."
        ::= { atmfVpcEntry 1 }

atmfVpcVpi OBJECT-TYPE
        SYNTAX  INTEGER (0..4095)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The VPI value of this Virtual Path Connection at the
                local ATM Interface."
        ::= { atmfVpcEntry 2 }

atmfVpcOperStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                        unknown(1),
                        end2endUp(2),
                        end2endDown(3),
                        localUpEnd2endUnknown(4),
                        localDown(5)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The present actual operational status of the VPC.

                A value of end2endUp(2) or end2endDown(3) would be
                used if the end-to-end status is known. If only local
                status information is available, a value of
                localUpEnd2endUnknown(4) or localDown(5) would be
                used."
        ::= { atmfVpcEntry 3 }

atmfVpcTransmitTrafficDescriptorType OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The type of traffic management, applicable to the
                transmit direction of this VPC. The type may indicate
                none, or a type with one or more parameters. These
                parameters are specified as a parameter vector, in the
                corresponding instances of the objects:
                        atmfVpcTransmitTrafficDescriptorParam1,
                        atmfVpcTransmitTrafficDescriptorParam2,
                        atmfVpcTransmitTrafficDescriptorParam3,
                        atmfVpcTransmitTrafficDescriptorParam4, and
                        atmfVpcTransmitTrafficDescriptorParam5."
```

```
        ::= { atmfVpcEntry 4 }

atmfVpcTransmitTrafficDescriptorParam1 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The first parameter of the transmit parameter vector
                for this VPC, used according to the value of
                atmfVpcTransmitTrafficDescriptorType."
        ::= { atmfVpcEntry 5 }

atmfVpcTransmitTrafficDescriptorParam2 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The second parameter of the transmit parameter vector
                for this VPC, used according to the value of
                atmfVpcTransmitTrafficDescriptorType."
        ::= { atmfVpcEntry 6 }

atmfVpcTransmitTrafficDescriptorParam3 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The third parameter of the transmit parameter vector
                for this VPC, used according to the value of
                atmfVpcTransmitTrafficDescriptorType."
        ::= { atmfVpcEntry 7 }

atmfVpcTransmitTrafficDescriptorParam4 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fourth parameter of the transmit parameter vector
                for this VPC, used according to the value of
                atmfVpcTransmitTrafficDescriptorType."
        ::= { atmfVpcEntry 8 }

atmfVpcTransmitTrafficDescriptorParam5 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fifth parameter of the transmit parameter vector
                for this VPC, used according to the value of
                atmfVpcTransmitTrafficDescriptorType."
        ::= { atmfVpcEntry 9 }

atmfVpcReceiveTrafficDescriptorType OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The type of traffic management, applicable to the
                traffic in the receive direction of this VPC. The type
                may indicate none, or a type with one or more
                parameters. These parameters are specified as a
                parameter vector, in the corresponding instances of
                the objects:
                        atmfVpcReceiveTrafficDescriptorParam1,
```

```
                        atmfVpcReceiveTrafficDescriptorParam2,
                        atmfVpcReceiveTrafficDescriptorParam3,
                        atmfVpcReceiveTrafficDescriptorParam4, and
                        atmfVpcReceiveTrafficDescriptorParam5."
        ::= { atmfVpcEntry 10 }

atmfVpcReceiveTrafficDescriptorParam1 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The first parameter of the receive parameter vector
                for this VPC, used according to the value of
                atmfVpcReceiveTrafficDescriptorType."
        ::= { atmfVpcEntry 11 }

atmfVpcReceiveTrafficDescriptorParam2 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The second parameter of the receive parameter vector
                for this VPC, used according to the value of
                atmfVpcReceiveTrafficDescriptorType."
        ::= { atmfVpcEntry 12 }

atmfVpcReceiveTrafficDescriptorParam3 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The third parameter of the receive parameter vector
                for this VPC, used according to the value of
                atmfVpcReceiveTrafficDescriptorType."
        ::= { atmfVpcEntry 13 }

atmfVpcReceiveTrafficDescriptorParam4 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fourth parameter of the receive parameter vector
                for this VPC, used according to the value of
                atmfVpcReceiveTrafficDescriptorType."
        ::= { atmfVpcEntry 14 }

atmfVpcReceiveTrafficDescriptorParam5 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fifth parameter of the receive parameter vector
                for this VPC, used according to the value of
                atmfVpcReceiveTrafficDescriptorType."
        ::= { atmfVpcEntry 15 }

atmfVpcQoSCategory OBJECT-TYPE
        SYNTAX  INTEGER {
                        other(1),
                        deterministic (2),
                        statistical (3),
                        unspecified (4)
                }
        ACCESS  read-only
```

```
        STATUS  obsolete
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 2.0
                of the UNI specification."
        ::= { atmfVpcEntry 16 }

atmfVpcTransmitQoSClass OBJECT-TYPE
        SYNTAX  INTEGER (0..255)
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfVpcEntry 17 }

atmfVpcReceiveQoSClass OBJECT-TYPE
        SYNTAX  INTEGER (0..255)
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfVpcEntry 18 }

atmfVpcBestEffortIndicator OBJECT-TYPE
        SYNTAX  TruthValue
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The object is examined when
                atmfVpcTransmitTrafficDescriptorType or
                atmfVpcReceiveTrafficDescriptorType for the associated
                connection is equal to atmfNoClpNoScr.
                If this object is set to false(2), the network is requested
                to apply the CBR.1 conformance definition. If this object
                is set to true(1), the network is requested to apply the
                UBR.1 conformance definition."
        ::= { atmfVpcEntry 19 }

atmfVpcServiceCategory OBJECT-TYPE
        SYNTAX  AtmServiceCategory
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The service category of this virtual path connection."
        ::= { atmfVpcEntry 20 }


--              The Virtual Path ABR Group
-- This group contains per-VPC information, support for which is optional.
--
-- Attributes of ABR Virtual Path connections

atmfVpcAbrTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfVpcAbrEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of operational parameters related to the ABR
                virtual path connections which cross this ATM
                Interface. There is one entry in this table for each
```

```
                ABR virtual path connection.

                Each virtual path connection represented
                in this table must also be represented by
                an entry in the atmfVpcTable."
        ::= { atmfVpcAbrGroup 1 }

atmfVpcAbrEntry OBJECT-TYPE
        SYNTAX  AtmfVpcAbrEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing information about a
                particular ABR virtual path connection."
        INDEX   { atmfVpcAbrPortIndex, atmfVpcAbrVpi }
        ::= { atmfVpcAbrTable 1 }

AtmfVpcAbrEntry ::=
        SEQUENCE {
                atmfVpcAbrPortIndex
                        INTEGER,
                atmfVpcAbrVpi
                        INTEGER,
                atmfVpcAbrTransmitIcr
                        INTEGER,
                atmfVpcAbrTransmitNrm
                        INTEGER,
                atmfVpcAbrTransmitTrm
                        INTEGER,
                atmfVpcAbrTransmitCdf
                        INTEGER,
                atmfVpcAbrTransmitRif
                        INTEGER,
                atmfVpcAbrTransmitRdf
                        INTEGER,
                atmfVpcAbrTransmitAdtf
                        INTEGER,
                atmfVpcAbrTransmitCrm
                        INTEGER
        }

atmfVpcAbrPortIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of identifying
                the ATM Interface over which this message was received."
        ::= { atmfVpcAbrEntry 1 }

atmfVpcAbrVpi OBJECT-TYPE
        SYNTAX  INTEGER (0..4095)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The VPI value of this ABR Virtual Path Connection at the
                local ATM Interface."
        ::= { atmfVpcAbrEntry 2 }

atmfVpcAbrTransmitIcr OBJECT-TYPE
        SYNTAX  INTEGER (0..16777215)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
```

```
                "Initial Cell Rate: This is the rate at which the
                source starts, both initially and after an idle period.
                The unit is cells per second.  The value must not be
                larger than PCR, and is usually lower."
        ::= { atmfVpcAbrEntry 3 }

atmfVpcAbrTransmitNrm OBJECT-TYPE
        SYNTAX  INTEGER {
                        nrm2(1),
                        nrm4(2),
                        nrm8(3),
                        nrm16(4),
                        nrm32(5),
                        nrm64(6),
                        nrm128(7),
                        nrm256(8)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of data cells a source may send
                for each forward RM-cell.  The default value is nrm32(5)."
        ::= { atmfVpcAbrEntry 4 }

atmfVpcAbrTransmitTrm OBJECT-TYPE
        SYNTAX  INTEGER {
                        trm0point78125(1),
                        trm1point5625(2),
                        trm3point125(3),
                        trm6point25(4),
                        trm12point5(5),
                        trm25(6),
                        trm50(7),
                        trm100(8)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Upper bound on the time between forward RM-cells for
                an active source (in milliseconds). The default value
                is trm100(8)."
        ::= { atmfVpcAbrEntry 5 }

atmfVpcAbrTransmitCdf OBJECT-TYPE
        SYNTAX  INTEGER {
                        cdf0(1),
                        cdfOneOver64(2),
                        cdfOneOver32(3),
                        cdfOneOver16(4),
                        cdfOneOver8(5),
                        cdfOneOver4(6),
                        cdfOneOver2(7),
                        cdfOne(8)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Cutoff Decrease Factor: This field controls the rate
                decrease associated with lost or delayed backward RM
                cells.  Larger values result in faster rate decrease.
                The default value is cdfOneOver16(4)."
        ::= { atmfVpcAbrEntry 6 }

atmfVpcAbrTransmitRif OBJECT-TYPE
```

```
        SYNTAX  INTEGER {
                        rifOneOver32768(1),
                        rifOneOver16384(2),
                        rifOneOver8192(3),
                        rifOneOver4096(4),
                        rifOneOver2048(5),
                        rifOneOver1024(6),
                        rifOneOver512(7),
                        rifOneOver256(8),
                        rifOneOver128(9),
                        rifOneOver64(10),
                        rifOneOver32(11),
                        rifOneOver16(12),
                        rifOneOver8(13),
                        rifOneOver4(14),
                        rifOneOver2(15),
                        rifOne(16)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Rate Increment Factor: Controls the rate at which the rate
                increases, when a backward RM-cell is received with CI=0 and
                NI=0.  Larger values lead to faster rate increase.  The default
                value is rifOneOver16(12)."
        ::= { atmfVpcAbrEntry 7 }

atmfVpcAbrTransmitRdf OBJECT-TYPE
        SYNTAX  INTEGER {
                        rdfOneOver32768(1),
                        rdfOneOver16384(2),
                        rdfOneOver8192(3),
                        rdfOneOver4096(4),
                        rdfOneOver2048(5),
                        rdfOneOver1024(6),
                        rdfOneOver512(7),
                        rdfOneOver256(8),
                        rdfOneOver128(9),
                        rdfOneOver64(10),
                        rdfOneOver32(11),
                        rdfOneOver16(12),
                        rdfOneOver8(13),
                        rdfOneOver4(14),
                        rdfOneOver2(15),
                        rdfOne(16)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Rate Decrease Factor: Controls the rate decrease
                which occurs when backward RM-cells with CI=1, are
                received.  Larger values lead to faster rate
                decrease.  The default value is rdfOneOver16(12)."
        ::= { atmfVpcAbrEntry 8 }

atmfVpcAbrTransmitAdtf OBJECT-TYPE
        SYNTAX  INTEGER (1..1023)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "ACR Decrease Time Factor: Time permitted between
                sending RM-cells, before the allowed rate (ACR) is
                decreased to ICR.  Range is 10 ms to 10.23 seconds.
                The unit is 10 milliseconds.  For example, the default
```

```
                value of 50 corresponds to a time factor of 500 ms.
                Larger values allow a source to retain its current
                rate longer, during periods of relative inactivity.
                The default is 50 (0.5 seconds)."
        ::= { atmfVpcAbrEntry 9 }

atmfVpcAbrTransmitCrm OBJECT-TYPE
        SYNTAX  INTEGER (0..8388608)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "RM Cells Before Cutoff: Limits the number of forward
                RM-cells which may be sent in the absence of received
                backward RM-cells."
        ::= { atmfVpcAbrEntry 10 }


--              The Virtual Channel Group
-- This group is mandatory for all ATM Interfaces.
--
-- Information concerning Virtual Channel Connections

atmfVccTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfVccEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of status and parameter information on the
                virtual channel connections which are visible at this
                ATM Interface. There is one entry in this table for
                each permanent virtual channel connection, including
                reserved VCCs that are supported; e.g., signalling,
                OAM flows, and ILMI, but not unassigned cells."
        ::= { atmfVccGroup 1 }

atmfVccEntry OBJECT-TYPE
        SYNTAX  AtmfVccEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing information about a
                particular virtual channel connection."
        INDEX   { atmfVccPortIndex, atmfVccVpi, atmfVccVci }
        ::= { atmfVccTable 1 }

AtmfVccEntry ::=
        SEQUENCE {
                atmfVccPortIndex
                        INTEGER,
                atmfVccVpi
                        INTEGER,
                atmfVccVci
                        INTEGER,
                atmfVccOperStatus
                        INTEGER,
                atmfVccTransmitTrafficDescriptorType
                        OBJECT IDENTIFIER,
                atmfVccTransmitTrafficDescriptorParam1
                        INTEGER,
                atmfVccTransmitTrafficDescriptorParam2
                        INTEGER,
                atmfVccTransmitTrafficDescriptorParam3
                        INTEGER,
                atmfVccTransmitTrafficDescriptorParam4
```

```
                        INTEGER,
                atmfVccTransmitTrafficDescriptorParam5
                        INTEGER,
                atmfVccReceiveTrafficDescriptorType
                        OBJECT IDENTIFIER,
                atmfVccReceiveTrafficDescriptorParam1
                        INTEGER,
                atmfVccReceiveTrafficDescriptorParam2
                        INTEGER,
                atmfVccReceiveTrafficDescriptorParam3
                        INTEGER,
                atmfVccReceiveTrafficDescriptorParam4
                        INTEGER,
                atmfVccReceiveTrafficDescriptorParam5
                        INTEGER,
                atmfVccQoSCategory
                        INTEGER,
                atmfVccTransmitQoSClass
                        INTEGER,
                atmfVccReceiveQoSClass
                        INTEGER,
                atmfVccBestEffortIndicator
                        TruthValue,
                atmfVccTransmitFrameDiscard
                        TruthValue,
                atmfVccReceiveFrameDiscard
                        TruthValue,
                atmfVccServiceCategory
                        AtmServiceCategory
        }

atmfVccPortIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of
                identifying the ATM Interface over which this message
                was received."
        ::= { atmfVccEntry 1 }

atmfVccVpi OBJECT-TYPE
        SYNTAX  INTEGER (0..4095)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The VPI value of this Virtual Channel Connection at
                the local ATM Interface. For virtual interfaces (i.e.
                Virtual Path Connections), this value has no meaning
                and is set to zero "
        ::= { atmfVccEntry 2 }

atmfVccVci OBJECT-TYPE
        SYNTAX  INTEGER (0..65535)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The VCI value of this Virtual Channel Connection at
                the local ATM Interface."
        ::= { atmfVccEntry 3 }

atmfVccOperStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                        unknown(1),
```

```
                        end2endUp(2),
                        end2endDown(3),
                        localUpEnd2endUnknown(4),
                        localDown(5)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The present actual operational status of the VCC. A
                value of end2endUp(2) or end2endUp(3) is used if the
                end to end status is known.

                If only local status is known a value of
                localUpEnd2endUnknown(4) or localDown(5) is used."
        ::= { atmfVccEntry 4 }

atmfVccTransmitTrafficDescriptorType OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The type of traffic management, applicable to the
                transmit direction of this VCC. The type may indicate
                none, or a type with one or more parameters. These
                parameters are specified as a parameter vector, in the
                corresponding instances of the objects:
                        atmfVccTransmitTrafficDescriptorParam1,
                        atmfVccTransmitTrafficDescriptorParam2,
                        atmfVccTransmitTrafficDescriptorParam3,
                        atmfVccTransmitTrafficDescriptorParam4, and
                        atmfVccTransmitTrafficDescriptorParam5."
        ::= { atmfVccEntry 5 }

atmfVccTransmitTrafficDescriptorParam1 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The first parameter of the transmit parameter vector
                for this VCC, used according to the value of
                atmfVccTransmitTrafficDescriptorType."
        ::= { atmfVccEntry 6 }

atmfVccTransmitTrafficDescriptorParam2 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The second parameter of the transmit parameter vector
                for this VCC, used according to the value of
                atmfVccTransmitTrafficDescriptorType."
        ::= { atmfVccEntry 7 }

atmfVccTransmitTrafficDescriptorParam3 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The third parameter of the transmit parameter vector
                for this VCC, used according to the value of
                atmfVccTransmitTrafficDescriptorType."
        ::= { atmfVccEntry 8 }

atmfVccTransmitTrafficDescriptorParam4 OBJECT-TYPE
```

```
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fourth parameter of the transmit parameter vector
                for this VCC, used according to the value of
                atmfVccTransmitTrafficDescriptorType."
        ::= { atmfVccEntry 9 }

atmfVccTransmitTrafficDescriptorParam5 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fifth parameter of the transmit parameter vector
                for this VCC, used according to the value of
                atmfVccTransmitTrafficDescriptorType."
        ::= { atmfVccEntry 10 }

atmfVccReceiveTrafficDescriptorType OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The type of traffic management, applicable to the
                traffic in the receive direction of this VCC. The type
                may indicate none, or a type with one or more
                parameters. These parameters are specified as a
                parameter vector, in the corresponding instances of
                the objects:
                        atmfVccReceiveTrafficDescriptorParam1,
                        atmfVccReceiveTrafficDescriptorParam2,
                        atmfVccReceiveTrafficDescriptorParam3,
                        atmfVccReceiveTrafficDescriptorParam4, and
                        atmfVccReceiveTrafficDescriptorParam5."
        ::= { atmfVccEntry 11 }

atmfVccReceiveTrafficDescriptorParam1 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The first parameter of the receive parameter vector
                for this VCC, used according to the value of
                atmfVccReceiveTrafficDescriptorType."
        ::= { atmfVccEntry 12 }

atmfVccReceiveTrafficDescriptorParam2 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The second parameter of the receive parameter vector
                for this VCC, used according to the value of
                atmfVccReceiveTrafficDescriptorType."
        ::= { atmfVccEntry 13 }

atmfVccReceiveTrafficDescriptorParam3 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The third parameter of the receive parameter vector
                for this VCC, used according to the value of
```

```
                atmfVccReceiveTrafficDescriptorType."
        ::= { atmfVccEntry 14 }

atmfVccReceiveTrafficDescriptorParam4 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fourth parameter of the receive parameter vector
                for this VCC, used according to the value of
                atmfVccReceiveTrafficDescriptorType."
        ::= { atmfVccEntry 15 }

atmfVccReceiveTrafficDescriptorParam5 OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The fifth parameter of the receive parameter vector
                for this VCC, used according to the value of
                atmfVccReceiveTrafficDescriptorType."
        ::= { atmfVccEntry 16 }

atmfVccQoSCategory OBJECT-TYPE
        SYNTAX  INTEGER {
                        other(1),
                        deterministic(2),
                        statistical(3),
                        unspecified(4)
                }
        ACCESS  read-only
        STATUS  obsolete
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 2.0
                of the UNI specification."
        ::= { atmfVccEntry 17 }

 atmfVccTransmitQoSClass OBJECT-TYPE
        SYNTAX  INTEGER (0..255)
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfVccEntry 18 }

atmfVccReceiveQoSClass OBJECT-TYPE
        SYNTAX  INTEGER (0..255)
        ACCESS  read-only
        STATUS  deprecated
        DESCRIPTION
                "This object should not be implemented except as
                required for backward compatibility with version 3.1
                of the UNI specification."
        ::= { atmfVccEntry 19 }

atmfVccBestEffortIndicator OBJECT-TYPE
        SYNTAX  TruthValue
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The object is examined when
```

```
                atmfVccTransmitTrafficDescriptorType or
                atmfVccReceiveTrafficDescriptorType for the associated
                connection is equal to atmfNoClpNoScr.
                If this object is set to false(2), the network is requested
                to apply the CBR.1 conformance definition. If this object
                is set to true(1), the network is requested to apply the
                UBR.1 conformance definition."
        ::= { atmfVccEntry 20 }

atmfVccTransmitFrameDiscard OBJECT-TYPE
        SYNTAX  TruthValue
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "If set to true(1), this object indicates that the network
                is requested to treat data for this connection (in the
                transmit direction) as frames (e.g. AAL5 CPCS_PDU's) rather
                than as individual cells.  While the precise
                implementation is network-specific, this treatment may
                for example involve discarding entire frames during
                congestion, rather than a few cells from many frames.
                The default value is false(2)."
        ::= { atmfVccEntry 21 }

atmfVccReceiveFrameDiscard OBJECT-TYPE
        SYNTAX  TruthValue
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "If set to true(1), this object indicates that the network
                is requested to treat data for this connection (in the
                receive direction) as frames (e.g. AAL5 CPCS_PDU's) rather
                than as individual cells.  While the precise
                implementation is network-specific, this treatment may
                for example involve discarding entire frames during
                congestion, rather than a few cells from many frames.
                The default value is false(2)."
        ::= { atmfVccEntry 22 }

atmfVccServiceCategory OBJECT-TYPE
        SYNTAX  AtmServiceCategory
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The service category of this virtual channel connection."
        ::= { atmfVccEntry 23 }


--              The Virtual Channel ABR Group
-- This group contains per-VCC information, support for which is optional.
--
-- Attributes of ABR Virtual Channel connections

atmfVccAbrTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfVccAbrEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of operational parameters related to the ABR
                virtual channel connections which cross this ATM
                Interface. There is one entry in this table for each
                ABR virtual channel connection.

                Each virtual channel connection represented
```

```
                in this table must also be represented by
                an entry in the atmfVccTable."
        ::= { atmfVccAbrGroup 1 }

atmfVccAbrEntry OBJECT-TYPE
        SYNTAX  AtmfVccAbrEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing information about a
                particular ABR virtual channel connection."
        INDEX   { atmfVccAbrPortIndex, atmfVccAbrVpi, atmfVccAbrVci }
        ::= { atmfVccAbrTable 1 }

AtmfVccAbrEntry ::=
        SEQUENCE {
                atmfVccAbrPortIndex
                        INTEGER,
                atmfVccAbrVpi
                        INTEGER,
                atmfVccAbrVci
                        INTEGER,
                atmfVccAbrTransmitIcr
                        INTEGER,
                atmfVccAbrTransmitNrm
                        INTEGER,
                atmfVccAbrTransmitTrm
                        INTEGER,
                atmfVccAbrTransmitCdf
                        INTEGER,
                atmfVccAbrTransmitRif
                        INTEGER,
                atmfVccAbrTransmitRdf
                        INTEGER,
                atmfVccAbrTransmitAdtf
                        INTEGER,
                atmfVccAbrTransmitCrm
                       INTEGER
        }

atmfVccAbrPortIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of identifying
                the ATM Interface over which this message was received."
        ::= { atmfVccAbrEntry 1 }

atmfVccAbrVpi OBJECT-TYPE
        SYNTAX  INTEGER (0..4095)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The VPI value of this ABR Virtual Channel Connection at the
                local ATM Interface. For virtual interfaces (i.e. Virtual Path
                Connections), this value has no meaning and is set to zero "
        ::= { atmfVccAbrEntry 2 }

atmfVccAbrVci OBJECT-TYPE
        SYNTAX  INTEGER (0..65535)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
```

```
                "The VCI value of this ABR Virtual Channel Connection at the
                local ATM Interface."
        ::= { atmfVccAbrEntry 3 }

atmfVccAbrTransmitIcr OBJECT-TYPE
        SYNTAX  INTEGER (0..16777215)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Initial Cell Rate: This is the rate at which the
                source starts, both initially and after an idle period.
                The unit is cells per second.  The value must not be
                larger than PCR, and is usually lower."
        ::= { atmfVccAbrEntry 4 }

atmfVccAbrTransmitNrm OBJECT-TYPE
        SYNTAX  INTEGER {
                        nrm2(1),
                        nrm4(2),
                        nrm8(3),
                        nrm16(4),
                        nrm32(5),
                        nrm64(6),
                        nrm128(7),
                        nrm256(8)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of data cells a source may send
                for each forward RM-cell.  The default value is nrm32(5)."
        ::= { atmfVccAbrEntry 5 }

atmfVccAbrTransmitTrm OBJECT-TYPE
        SYNTAX  INTEGER {
                        trm0point78125(1),
                        trm1point5625(2),
                        trm3point125(3),
                        trm6point25(4),
                        trm12point5(5),
                        trm25(6),
                        trm50(7),
                        trm100(8)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Upper bound on the time between forward RM-cells for
                an active source (in milliseconds). The default value
                is trm100(8)."
        ::= { atmfVccAbrEntry 6 }

atmfVccAbrTransmitCdf OBJECT-TYPE
        SYNTAX  INTEGER {
                        cdf0(1),
                        cdfOneOver64(2),
                        cdfOneOver32(3),
                        cdfOneOver16(4),
                        cdfOneOver8(5),
                        cdfOneOver4(6),
                        cdfOneOver2(7),
                        cdfOne(8)
                }
        ACCESS  read-only
```

```
        STATUS  mandatory
        DESCRIPTION
                "Cutoff Decrease Factor: This field controls the rate
                decrease associated with lost or delayed backward RM
                cells.  Larger values result in faster rate decrease.
                The default value is cdfOneOver16(4)."
        ::= { atmfVccAbrEntry 7 }

atmfVccAbrTransmitRif OBJECT-TYPE
        SYNTAX  INTEGER {
                        rifOneOver32768(1),
                        rifOneOver16384(2),
                        rifOneOver8192(3),
                        rifOneOver4096(4),
                        rifOneOver2048(5),
                        rifOneOver1024(6),
                        rifOneOver512(7),
                        rifOneOver256(8),
                        rifOneOver128(9),
                        rifOneOver64(10),
                        rifOneOver32(11),
                        rifOneOver16(12),
                        rifOneOver8(13),
                        rifOneOver4(14),
                        rifOneOver2(15),
                        rifOne(16)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Rate Increment Factor: Controls the rate at which the rate
                increases, when a backward RM-cell is received with CI=0 and
                NI=0.  Larger values lead to faster rate increase.  The default
                value is rifOneOver16(12)."
        ::= { atmfVccAbrEntry 8 }

atmfVccAbrTransmitRdf OBJECT-TYPE
        SYNTAX  INTEGER {
                        rdfOneOver32768(1),
                        rdfOneOver16384(2),
                        rdfOneOver8192(3),
                        rdfOneOver4096(4),
                        rdfOneOver2048(5),
                        rdfOneOver1024(6),
                        rdfOneOver512(7),
                        rdfOneOver256(8),
                        rdfOneOver128(9),
                        rdfOneOver64(10),
                        rdfOneOver32(11),
                        rdfOneOver16(12),
                        rdfOneOver8(13),
                        rdfOneOver4(14),
                        rdfOneOver2(15),
                        rdfOne(16)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Rate Decrease Factor: Controls the rate decrease
                which occurs when backward RM-cells with CI=1, are
                received.  Larger values lead to faster rate
                decrease.  The default value is rdfOneOver16(12)."
        ::= { atmfVccAbrEntry 9 }
```

```
atmfVccAbrTransmitAdtf OBJECT-TYPE
        SYNTAX  INTEGER (1..1023)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "ACR Decrease Time Factor: Time permitted between
                sending RM-cells, before the allowed rate (ACR) is
                decreased to ICR.  Range is 10 ms to 10.23 seconds.
                The unit is 10 milliseconds.  For example, the default
                value of 50 corresponds to a time factor of 500 ms.
                Larger values allow a source to retain its current
                rate longer, during periods of relative inactivity.
                The default is 50 (0.5 seconds)."
        ::= { atmfVccAbrEntry 10 }

atmfVccAbrTransmitCrm OBJECT-TYPE
        SYNTAX  INTEGER (0..8388608)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "RM Cells Before Cutoff: Limits the number of forward
                RM-cells which may be sent in the absence of received
                backward RM-cells."
        ::= { atmfVccAbrEntry 11 }


-- Traps
atmfVpcChange TRAP-TYPE
        ENTERPRISE      atmForum
        VARIABLES       { atmfVpcPortIndex, atmfVpcVpi, atmfVpcOperStatus }
        DESCRIPTION
                "An atmfVpcChange trap indicates that a permanent VPC has been
                added or deleted at this ATM Interface or that the attributes
                of an existing VPC have been modified. The variables
                included in the trap identify the VPI value of the
                reconfigured VPC at this ATM Interface."
        ::= 1

atmfVccChange TRAP-TYPE
        ENTERPRISE      atmForum
        VARIABLES       { atmfVccPortIndex, atmfVccVci, atmfVccVpi,
                          atmfVccOperStatus }
        DESCRIPTION
                "An atmfVccChange trap indicates that a permanent VCC has been
                added or deleted at this ATM Interface or that the attributes
                of an existing VCC have been modified.. The variables
                included in the trap identify the VCI and VPI values
                of the reconfigured VCC at this ATM
                Interface."
        ::= 2

END
```

9. Address Registration MIB

This section specifies the procedures for address registration at the UNI. These procedures are specified as an extension to the ILMI.

Equipment at the Private UNI must support the Address Registration procedures described in this section in their entirety.

Equipment at the Public UNI must support the procedures for the Address Registration Admin Group described in this section, and may support the Address Registration procedures described in this section in their entirety.

9.1 Overview

In order to establish an ATM connection at the UNI, both the user and the network must know the ATM address(es) which are in effect at that UNI. These ATM addresses can then be used in Calling Party Number information elements of signalling messages sent by the user, and in Called Party Number information elements of signalling messages sent to the user. The address registration procedures in this section provide the means for the dynamic exchange of addressing information between the user and the network at the UNI, at initialization and at other times as required. Through this dynamic exchange the user and network can agree on the ATM address(es) in effect.

As specified in [UNI4.0], the Private ATM Address Structure consists of multiple fields. Two of these fields, the End System Identifier (ESI) and the Selector (SEL) fields form the “user part” and are supplied by the user-side IME. All other fields form a “network prefix” for ATM addresses in that they typically have the same value for all ATM addresses on the same ATM UNI; the value of the network prefix is supplied by the network side of the UNI. The network-side IME is allowed to supply multiple network prefixes for use at a single UNI; however, it is expected that just one will normally be supplied. An ATM address for a terminal on the user side of a Private UNI is obtained by appending values for the ESI and SEL fields to (one of) the network prefix(es) for that UNI.

For the purposes of address registration, the value of the SEL field is irrelevant, i.e., one user part is a duplicate of another if they have the same ESI value even if they have different SEL values.

ATM addresses in Public networks either have the same Private ATM Address Structure as Private networks, or else they are Native E.164 addresses (see [UNI4.0]). For Native E.164 addresses, the network-side IME supplies the whole ATM address. In effect, the network prefix is the Native E.164 address and can only be validly combined with a null user part. In this situation, the use of multiple network prefixes at a single UNI occurs whenever multiple Native E.164 addresses are used at that UNI.

9.2 Capabilities

The address registration procedures in this section provide the following capabilities:

- Initialization-time exchange of addressing information,
- Restrictions on network-prefix/user-part combinations,

- Acceptance of Unassigned Network-Prefixes,
- Rejection of unacceptable values, either rejection of a specific network prefix by the user, or of a specific user part by the network,
- Dynamic addition/deletion of additional network prefixes and user parts,
- De-registration of addresses when ILMI Connectivity is lost,
- An indication of support or non-support for address registration at an interface.

The following sections describe each capability in more detail.

9.2.1 Initialization-time Exchange of Addressing Information

These procedures allow the user and the network to exchange addressing information to allow ATM addresses to be registered at the UNI and used in signalling messages. This includes the capability for the network to support more than one network prefix (e.g., in a transitional period to allow both an old and a new network prefix value), and for the user to support more than one user part.

9.2.2 Restrictions on Network-Prefix/User-Part Combinations

Not all combinations of network prefix and user part may be valid. For example, if the network prefix is an Native E.164 address (using the E.164 numbering plan), then no user part is necessary or allowed. Alternatively, at a Public UNI which supports Address Registration and connects a Private ATM network and a Public ATM network, the user (the Private ATM network) may wish to restrict combinations of network prefixes offered by the Public network and user parts that it registers (e.g., to simplify its routing task).

9.2.3 Acceptance of Unassigned Network-Prefixes

The network may accept registration of ATM Addresses with network prefixes which were not supplied by the network. This may occur for well-known network prefixes or Group Addresses.

9.2.4 Rejection of Unacceptable Values

Certain user part values, as specified by the user, may be unacceptable to the network. An example would be if the user attempted to register a user part which is already registered on a different UNI. In this case, the network may wish to block registration of the duplicate. Similarly, a user may wish to block registration of a network prefix supplied by the network. For example, at a Public UNI which supports Address Registration and supports E.164 addresses, both using the Native E.164 numbering plan and using NSAP encoding, the user might wish to block registration one of the address formats.

9.2.5 Dynamic Addition/Deletion

The list of network prefixes and user parts, and their valid combinations may change over time during the operation of the UNI. For example, on the user side of a private UNI, new user parts may appear or disappear when (individually addressable) components are connected to or disconnected from the access device. On the network side, the list of valid network prefixes may change due to topology changes.

9.2.6 De-registration on ILMI Connectivity Lost Condition

In order for a user device using the Private ATM Address structure to be unplugged from one UNI and plugged in to another UNI on the same network, the user-part(s) registered by the device must be de-registered from the first UNI and re-registered on the second. In order to rapidly change address registration locations, the addresses registered at the UNI are de-registered when ILMI Connectivity is lost.

9.2.7 Indication of Non-support for Address Registration

In some instances, in particular in some public networks, address registration may not be supported. In order to maximize flexibility and allow for different configurations, indication of support or non-support for address registration may be configured by network management. When support is indicated on both sides of an interface, address registration procedures will be affected as described in the text, otherwise they will be bypassed.

9.3 General Description of Procedures

For the exchange of addressing information, two MIB tables are defined, one to contain network prefixes, and the other to contain registered ATM addresses. The Network Prefix table contains one entry for each network prefix. The Address table contains one entry for each registered ATM address. The MIB definitions of these tables are given in section 9.6. The basic approach of the procedures is as follows.

For network prefixes, it is the network-side IME which supplies the values for the user-side IME to accept or reject. Thus, it is the user-side IME which implements the Network Prefix table, and the network-side IME which issues SetRequest messages to create/delete entries in the table in order to register/de-register network prefixes.

For registered addresses, it is the user-side IME which supplies the values for the network-side IME to accept or reject. Thus, it is the network-side IME which implements the Address table, and the user-side IME which issues SetRequest messages to create/delete entries in the table in order to register/de-register ATM addresses.

At initialization, the registration of network prefixes occurs first. Next, the user-side IME combines each of the user parts it wishes to use with one or more of the registered network prefixes to form a set of ATM addresses. The user-side IME then registers these addresses.

After initialization, the network-side IME issues SetRequest messages to create/delete entries in the Network Prefix table as and when new network prefixes need to be added or existing network prefixes deleted. Similarly, the user-side IME issues SetRequest messages to create/delete entries in the Address table as and when new ATM addresses need to be registered or existing ATM addresses de-registered. If and when the IME loses ILMI Connectivity, all addresses are de-registered.

9.4 Management Information Base

Address registration is defined as an extension to the ILMI through the definition of three additional MIB groups: the NetPrefix group, the Address group, and the Address Registration Admin group.

9.4.1 NetPrefix Group (CR)

The NetPrefix Group (atmfNetPrefixGroup) is required to be implemented by the IME on the user side of the Private UNI, and may be implemented by the IME on the user side of the Public UNI. This group consists of one MIB table, the Network Prefix Table, indexed by the ATM Interface Index and by the value of a network prefix. The information in this group is:

- Interface Index
- Network Prefix
- Network Prefix Status

9.4.1.1 Interface Index (R)

The Interface Index object (atmfNetPrefixPort) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

9.4.1.2 Network Prefix (R)

The Network Prefix object (atmfNetPrefixPrefix) has the value of a network prefix. In the case of private ATM address structures, the network prefix is the first 13 octets of the address which includes the AFI, IDI and HO-DSP fields. In the case of Native E.164 address structures (see [UNI4.0]), the network prefix is the entire E.164 address encoded in 8 octets, as if it were an E.164 IDP in a private ATM address structure.

9.4.1.3 Network Prefix Status (R)

The Network Prefix Status object (atmfNetPrefixStatus) provides an indication of the validity of a network prefix at this ATM Interface. To configure a new network prefix, the network-side IME uses a SetRequest to set the Network Prefix Status object for that prefix to be valid. To delete an existing network prefix, the network-side IME uses a SetRequest to set the Network Prefix Status object for that prefix to be invalid.

9.4.2 Address Group (CR)

The Address group (atmfAddressGroup) is required to be implemented by the IME on the network side of the Private UNI, and may be implemented by the IME on the network side of the Public UNI. This group consists of one MIB table, the Address Table, which is indexed by the ATM Interface Index and by the value of an ATM address. The information in this group is:

- Interface Index
- ATM Address
- ATM Address Status
- ATM Address Organizational Scope Indication

9.4.2.1 Interface Index (R)

The Interface Index object (atmfAddressPort) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

9.4.2.2 ATM Address (R)

The ATM Address object (atmfAddressAtmAddress) has the value of an ATM address. In the case of private ATM address structures, the ATM address is the entire 20 octets of the address which includes the AFI, IDI, HO-DSP, ESI and SEL fields. In the case of Native E.164 address structures (see [UNI4.0]), the ATM address is the entire E.164 address encoded in 8 octets, as if it were an E.164 IDP in a private ATM address structure (in this case, the network prefix and ATM address are identical and the user part is null).

9.4.2.3 ATM Address Status (R)

The ATM Address Status object (atmfAddressStatus) provides an indication of the validity of an ATM address at this ATM Interface. To configure a new ATM address, the user-side IME uses a SetRequest to set the ATM Address Status object for that address to be valid. To delete an existing ATM address, the user-side IME uses a SetRequest to set the ATM Address Status object for that address to be invalid.

9.4.2.4 ATM Address Organizational Scope (R)

The ATM Address Organizational Scope object (atmfAddressOrgScope) indicates the organizational scope for the associated address. An organizational scope may apply to individual addresses as well as group addresses. The associated address shall not be distributed outside the indicated scope. Refer to Annex 6.0 of [UNI4.0] for guidelines regarding the use of organizational scopes.

This organization hierarchy will be mapped to the ATM network's routing hierarchy, such as PNNI's routing level [PNNI1.0], and the mapping shall be configurable in the ATM nodes. Use of this object in a public network is for further study.

Implementation of the object atmfAddressOrgScope is required to support the ATM anycast capability (see [UNI4.0]).

9.4.3 Address Registration Admin Group (R)

The Address Registration Admin group (atmfAddressRegistrationAdminGroup) is required to be implemented by all IMEs. This group consists of one MIB table, the Address Registration Admin Table, indexed by the ATM Interface Index. The information in this group is:

- Interface Index
- Address Registration Admin Status

9.4.3.1 Interface Index (R)

The Interface Index object (atmfAddressRegistrationAdminIndex) is the same as that for the Physical interface as defined in section 8.2.2.1. It is implicitly the local ATM Interface.

9.4.3.2 Address Registration Admin Status (R)

The Address Registration Admin Status object (atmfAddressRegistrationAdminStatus) provides an indication of support for the Prefix and Address Groups. The Prefix and Address Groups are supported only if the Address Registration Admin Status object on both sides of the UNI indicate such support. Support for the Prefix and Address Groups must not be determined implicitly by the interface type.

Because the Prefix and Address Groups are mandatory at Private UNIs, implementations may choose to always set the Address Registration Admin Status object to indicate support.

9.5 Procedures

9.5.1 Network-Side IME

Upon its own restart and before sending the standard coldStart trap, the IME initializes the Address Table to be empty. It then sends a coldStart trap and invokes its ILMI Connectivity and Automatic Configuration Procedures. If the IME is determined to be the network side of a Public or Private UNI, the IME sends an ILMI GetNextRequest message to read the first instance of the Network Prefix Status object from the user-side IME. If the response does not indicate that the Network Prefix table is empty, then this procedure is restarted by retransmitting the coldStart trap and re-invoking the ILMI Connectivity and Automatic Configuration Procedures.

The network-side IME also initializes the Address Table to be empty upon receipt of a coldStart trap from the user-side IME. Receipt of a coldStart trap from the user-side IME indicates that the user-side has restarted and initialized its Network Prefix Table to be empty (see 9.5.2).

In either case, the user-side IME's Network Prefix table will now be empty. The network-side IME may issue a GetRequest to read the atmfAddressRegistrationAdminStatus object from the user-side IME. The user-side IME indicates that it supports Address Registration by returning supported(1), and that it does not support Address Registration by returning unsupported(2). If the user-side IME returns a noSuchName error (indicating that it is a pre-ILMI 4.0 IME), the network-side IME must assume that the user-side IME supports Address Registration.

If the network-side IME supports Address Registration, but the response to the GetRequest indicates that the user-side IME does not, it is suggested that the network-side IME indicate an alarm condition to the network manager indicating UNI misconfiguration. Further, as an optimization, the network-side IME may refrain from the following procedure to register Net Prefixes. No harm is done by attempting registration anyway, as the user-side IME should simply return noSuchName errors to any such registration requests.

Assuming the optimization described above is not utilized, the network-side IME may now issue any number of SetRequests to register its Net Prefix(es), e.g., SetRequest { atmfNetPrefixStatus.port.prefix=valid(1)}

After it has transmitted a coldStart trap, including the period before which it has validated that the user-side IME's Net Prefix table has been emptied, the network-side IME must properly process all SNMP messages including GetRequests, GetNextRequests, and SetRequests for the Address Table.

If at any time, the network-side IME wishes to register an additional network prefix, it issues an appropriate SetRequest, e.g.,

SetRequest { atmfNetPrefixStatus.port.prefix=valid(1) }

If at any time, the network-side IME wishes to de-register a network prefix, it issues an appropriate SetRequest, e.g.,

SetRequest { atmfNetPrefixStatus.port.prefix=invalid(2) }

If at any time, the network-side IME wishes to check the consistency of the set of network prefixes currently registered, it issues ILMI messages (e.g., GetNextRequests) to read all instances of the Network Prefix Status object from the user-side IME and checks that the set of those which have a valid status is the correct set.

Upon receipt of a SetRequest setting an instance of the Address Status object to be valid, the network-side IME validates the referenced address. The validation may consist of verifying the network prefix, the organizational scope, and preventing registration of duplicate individual addresses. If the validation fails, it responds with a GetResponse containing a badValue error. If the validation succeeds, if responds with a GetResponse indicating noError, and if the address is not currently registered, it is registered and the Address table is updated.

On a SetRequest which specifies an entry which is identical to an address entry existing with the IME, the IME responds with a noError.

If the user-side IME does not specify a value for the atmfAddressOrgScope object, the network shall set the value of this object to localNetwork(1), if the registered address is an ATM group address, or to global(15), if the registered address is an individual address.

If the user-side IME attempts to change the value for the atmfAddressOrgScope object to an invalid value, the network-side IME responds with a GetResponse containing a badValue error, but leaves the old value in effect.

Upon receipt of a SetRequest setting an instance of the Address Status object to be invalid, the network-side IME checks if the referenced address is currently registered. If not, it responds with a GetResponse containing a noSuchName error. If the address is currently registered, then it responds with a GetResponse indicating noError, the address is de-registered and the corresponding atmfAddressEntry is removed from the Address table as soon as possible.

When a user side IME de-registers an ATM address, the network side call control entity must NOT clear any connections/calls associated with the de-registered address.

During operation, if the network-side IME loses ILMI Connectivity, it de-registers all addresses. Procedures for maintaining ILMI Connectivity are specified in section 8.3.1.

9.5.2 User-Side IME

Upon its own restart and before sending the standard coldStart trap, the IME initializes the Network Prefix Table to be empty. It then sends a coldStart trap and invokes its ILMI Connectivity and Automatic Configuration Procedures. If the IME is determined to be the user side of a Public or Private UNI, the IME sends an ILMI GetNextRequest message to read the first instance of the ATM Address Status object from the network-side IME. If the response does not indicate that the ATM Address table is empty, then this procedure is restarted by retransmitting the coldStart trap and re-invoking the ILMI Connectivity and Automatic Configuration Procedures.

The user-side IME also initializes the Network Prefix Table to be empty upon receipt of a coldStart trap from the network-side IME. Receipt of a coldStart trap from the network-side IME indicates that the network-side has restarted and initialized its Address Table to be empty (see 9.5.1).

In either case the network-side IME's ATM Address table will now be empty. The user-side IME may issue a GetRequest to read the atmfAddressRegistrationAdminStatus object from the network-side IME. The network-side IME indicates that it supports Address Registration by returning supported(1), and that it does not support Address Registration by returning unsupported(2). If the network-side IME returns a noSuchName error (indicating that it is a pre-ILMI 4.0 IME), the user-side IME must assume that the network-side IME supports Address Registration.

If the user-side IME supports Address Registration, but the network-side IME does not, it is suggested that the user-side IME indicate an alarm condition to the system administrator indicating UNI misconfiguration. Further, as an optimization, the user-side IME may refrain from the following procedure to register ATM Addresses. No harm is done by attempting registration anyway, as the network-side IME should simply return noSuchName errors to any such registration requests.

Assuming the optimization described above is not utilized, the user-side IME may now issue any number of SetRequests to register its ATM Address(es), e.g.,

SetRequest { atmfAddressStatus.port.address=valid(1)}

After it has transmitted a coldStart trap, including the period before which it has validated that the network-side IME's Address Table has been emptied, the user-side IME must properly process all SNMP messages including GetRequests, GetNextRequests, and SetRequests for the Net Prefix Table.

Upon receipt of a SetRequest setting an instance of the Network Prefix Status object to be valid, the user-side IME validates the SetRequest. If the validation fails, it responds with a GetResponse containing the appropriate error. If the validation succeeds, it responds with a GetResponse indicating noError, and if the network prefix is one not currently registered, the prefix is registered and the Network Prefix table is updated. For any address which the user-side IME wishes to register using the new prefix, it forms the address in one of two ways: by appending the ESI and SEL values to the prefix or, for Native E.164 addresses, by appending a null user part. It then issues an appropriate SetRequest, e.g.,

SetRequest { atmfAddressStatus.port.address=valid(1) }

Upon receipt of a SetRequest setting an instance of the Network Prefix Status object to be invalid, the user-side IME checks if the prefix is currently registered. If not, it responds with a GetResponse containing a noSuchName error. If the prefix is currently registered, then it responds with a GetResponse indicating noError, the prefix is de-registered and the corresponding AtmfNetPrefixEntry is removed from the Network Prefix table as soon as possible.

If at any time, the user-side IME wishes to register an additional address having the Private ATM Address structure, it forms the address by appending the ESI and SEL values to one of the registered prefixes and issues an appropriate SetRequest, e.g.,

SetRequest { atmfAddressStatus.port.address=valid(1) }

If at any time, the user-side IME wishes to de-register an address, it issues an appropriate SetRequest, e.g.,

SetRequest { atmfAddressStatus.port.address=invalid(2) }

Upon de-registering a prefix, the user-side IME should immediately de-register all addresses that it registered by combining that prefix with an ESI and SEL value.

If the user-side IME wishes to check periodically that (at least one of) its address(es) is still registered (e.g., that it has not been unplugged from one UNI and plugged into another), then it may wish to use the Change of Attachment Point Procedure.

If at any time, the user-side IME wishes to check for consistency of all ATM addresses currently registered, it uses ILMI messages (e.g., GetNextRequests) to read all instances of the Address Status object from the network-side IME and checks that the set of those with valid status is the correct set. (Note that when a single address is registered, this consistency check is equivalent to checking just the first address; when multiple addresses are registered, this full consistency check is expected to be needed much less frequently than checking just the first address.)

The following additional procedures for registration of organizational scope apply when the registered address is an ATM Group address, and optionally when the registered address is an individual address as well.

For any address that the user side wishes to register with a specified organizational scope, the user side IME may issue an appropriate SetRequest any time after the coldStart trap has taken effect, i.e. after making sure that the ATM Address table is empty. The user-side IME registers the address and its organizational scope by issuing a SetRequest, e.g.:

SetRequest { atmfAddressStatus.port.address=valid(1),
atmfAddressOrgScope.port.address=localNetwork(1)}

To change the values of the organizational scope object of a registered address, the user-side IME re-registers the desired address value by issuing a SetRequest, e.g.:

SetRequest { atmfAddressOrgScope.port.address=localNetwork(1)}

9.5.3 Retransmission of SetRequest Messages

For the above procedures, an IME should retransmit a SetRequest message if it does not receive a GetResponse message within a time-out period. There are two possibilities when a GetResponse is not received, e.g. due to a transmission error/loss, either: A) the SetRequest was processed and the generated GetResponse was lost, or B) the SetRequest was lost before it was processed. When registering a network prefix/address, the retransmission of the appropriate SetRequest message produces the same result no matter which of A or B occurred. When de-registering a network prefix/address, the GetResponse to a retransmission of the appropriate SetRequest message will have an error-status of either noError, noSuchName, or some other error. The noError status indicates that situation B existed, but the de-registration is now complete. The noSuchName status indicates that situation A existed, i.e. the prefix/address was already de-registered. Some other error indicates that the de-registration cannot be completed for some other reason.

9.6 MIB Definitions

```
ATM-FORUM-ADDR-REG  DEFINITIONS ::=  BEGIN

IMPORTS
        atmfNetPrefixGroup,
        atmfAddressGroup,
        atmfAddressRegistrationAdminGroup,
        AtmAddress,
        NetPrefix                          FROM ATM-FORUM-TC-MIB
        OBJECT-TYPE                        FROM RFC-1212;


--                  The NetPrefix Group
--
-- The Network Prefix Table is implemented by the user-side IME.

atmfNetPrefixTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfNetPrefixEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table implemented by the user-side IME, containing the
                network-prefix(es) for ATM-layer addresses in effect on
                the user side of the UNI."
        ::= { atmfNetPrefixGroup 1 }

atmfNetPrefixEntry OBJECT-TYPE
        SYNTAX  AtmfNetPrefixEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "Information about a single network-prefix for
                ATM-layer addresses in effect on the user-side IME.
```

```
                Note that the index variable atmNetPrefixPrefix is a
                variable-length string, and as such the rule for
                variable-length strings in section 4.1.6 of RFC 1212
                applies."

        INDEX   { atmfNetPrefixPort, atmfNetPrefixPrefix }
        ::= { atmfNetPrefixTable 1 }

AtmfNetPrefixEntry ::=
        SEQUENCE {
                atmfNetPrefixPort
                        INTEGER,
                atmfNetPrefixPrefix
                        NetPrefix,
                atmfNetPrefixStatus
                        INTEGER
        }

atmfNetPrefixPort OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A unique value which identifies the UNI port for
                which the network prefix for ATM addresses is in
                effect.  The value of 0 has the special meaning of
                identifying the local UNI."
        ::= { atmfNetPrefixEntry 1 }

atmfNetPrefixPrefix  OBJECT-TYPE
        SYNTAX  NetPrefix
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "The network prefix for ATM addresses which is in
                effect on the user side of the ATM UNI port."
        ::= { atmfNetPrefixEntry 2 }

atmfNetPrefixStatus  OBJECT-TYPE
        SYNTAX  INTEGER { valid(1), invalid(2) }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "An indication of the validity of the network prefix
                for ATM addresses on the user side of the UNI port.
                To configure a new network prefix in this table, the
                network-side IME must set the appropriate instance of this
                object to the value valid(1).  To delete an existing
                network prefix in this table, the network-side IME must
                set the appropriate instance of this object to the
                value invalid(2).

                If circumstances occur on the user-side IME which cause a
                prefix to become invalid, the user-side IME modifies the
                value of the appropriate instance of this object to invalid(2).

                Whenever the value of this object for a particular
                prefix becomes invalid(2), the conceptual row for that
                prefix may be removed from the table at any time,
                either immediately or subsequently."
        ::= { atmfNetPrefixEntry 3 }


--                      The Address Group
```

```
--
-- The Address Table is implemented by the network-side IME.

atmfAddressTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfAddressEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table implemented by the network-side IME, containing the
                ATM-layer addresses in effect on the user side of the UNI."
        ::= { atmfAddressGroup 1 }

atmfAddressEntry OBJECT-TYPE
        SYNTAX  AtmfAddressEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "Information about a single ATM-layer address in effect
                on the user-side IME.  Note that the index variable
                atmAddressAtmAddress is a variable-length string, and as
                such the rule for variable-length strings in section
                4.1.6 of RFC 1212 applies."
        INDEX   { atmfAddressPort, atmfAddressAtmAddress }
        ::= { atmfAddressTable 1 }

AtmfAddressEntry ::=
        SEQUENCE {
                atmfAddressPort
                        INTEGER,
                atmfAddressAtmAddress
                        AtmAddress,
                atmfAddressStatus
                        INTEGER,
                atmfAddressOrgScope
                        INTEGER
        }

atmfAddressPort OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A unique value which identifies the UNI port for
                which the ATM address is in effect.  The value of 0
                has the special meaning of identifying the local UNI."
        ::= { atmfAddressEntry 1 }

atmfAddressAtmAddress  OBJECT-TYPE
        SYNTAX  AtmAddress
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "The ATM address which is in effect on the user side
                of the ATM UNI port."
        ::= { atmfAddressEntry 2 }

atmfAddressStatus  OBJECT-TYPE
        SYNTAX  INTEGER { valid(1), invalid(2) }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "An indication of the validity of the ATM address at
                the user side of the UNI port.  To configure a new
                address in this table, the user-side IME must set the
```

```
                appropriate instance of this object to the value
                valid(1).  To delete an existing address in this table,
                the user-side IME must set the appropriate instance of
                this object to the value invalid(2).

                If circumstances occur on the network-side IME which cause
                an address to become invalid, the network-side IME
                modifies the value of the appropriate instance of this
                object to invalid(2).

                Whenever the value of this object for a particular
                address becomes invalid(2), the conceptual row for
                that address may be removed from the table at any
                time, either immediately or subsequently."
        ::= { atmfAddressEntry 3 }

atmfAddressOrgScope OBJECT-TYPE
        SYNTAX  INTEGER {
                localNetwork(1),
                localNetworkPlusOne(2),
                localNetworkPlusTwo(3),
                siteMinusOne(4),
                intraSite(5),
                sitePlusOne(6),
                organizationMinusOne(7),
                intraOrganization(8),
                organizationPlusOne(9),
                communityMinusOne(10),
                intraCommunity(11),
                communityPlusOne(12),
                regional(13),
                interRegional(14),
                global(15)
        }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "This object indicates the organizational
                scope for the referenced address. The information of
                the referenced address shall not be distributed
                outside the indicated scope. If the user-side IME does
                not specify a value for the atmfAddressOrgScope object,
                the network shall set the value of this object to
                localNetwork(1), if the registered address is an ATM group
                address, or to global(15), if the registered address is
                an individual address.  Refer to Annex 6.0
                of ATM Forum UNI Signalling 4.0 for guidelines regarding
                the use of organizational scopes.

                This organization hierarchy may be mapped to ATM network's
                routing hierarchy such as PNNI's routing level and
                the mapping shall be configurable in
                nodes. Use of this object in a public network is for
                further study.
                The default values for organizational scope are
                localNetwork(1) for ATM group addresses, and global(15)
                for individual addresses."
        ::= { atmfAddressEntry 4 }


--                      The Address Registration Admin Group
--
-- The Address Registration Admin Table is mandatory for all IMEs.
```

```
atmfAddressRegistrationAdminTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfAddressRegistrationAdminEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table of Address Registration administrative
                information for the ATM Interface."
    ::= { atmfAddressRegistrationAdminGroup 1 }

atmfAddressRegistrationAdminEntry OBJECT-TYPE
        SYNTAX  AtmfAddressRegistrationAdminEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the table, containing Address
                Registration administrative information for the ATM
                Interface."
        INDEX   { atmfAddressRegistrationAdminIndex }
        ::= { atmfAddressRegistrationAdminTable 1 }


AtmfAddressRegistrationAdminEntry ::=
        SEQUENCE {
                atmfAddressRegistrationAdminIndex
                        INTEGER,
                atmfAddressRegistrationAdminStatus
                        INTEGER
        }

atmfAddressRegistrationAdminIndex OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of
                identifying the ATM Interface over which this message
                was received."
        ::= { atmfAddressRegistrationAdminEntry 1 }

atmfAddressRegistrationAdminStatus OBJECT-TYPE
        SYNTAX  INTEGER { supported(1), unsupported(2) }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An indication of whether or not Address Registration
                is supported on this ATM Interface. Supported(1)
                indicates that this ATM Interface supports address
                registration. Unsupported(2) indicates that this ATM
                Interface does not support address registration."
        ::= { atmfAddressRegistrationAdminEntry 2 }
END
```

10. Service Registry MIB

This specification extends the ATM Interface MIB to provide a general-purpose service registry for locating ATM network services such as the LAN Emulation Configuration Server (LECS) and the ATM Name Server (ANS).

10.1 MIB Definitions

```
ATM-FORUM-SRVC-REG DEFINITIONS ::= BEGIN

IMPORTS
        atmfSrvcRegTypes,
        atmfSrvcRegistryGroup,
        AtmAddress                      FROM ATM-FORUM-TC-MIB
        OBJECT-TYPE                     FROM RFC-1212;


--              Object Identifier definitions

-- The following values are defined for use as possible values
-- of the atmfSrvcRegServiceID object.

-- LAN Emulation Configuration Server (LECS)
atmfSrvcRegLecs         OBJECT IDENTIFIER ::= { atmfSrvcRegTypes 1 }
-- When atmfSrvcRegServiceID has a value of atmfSrvcRegLecs,
-- the value of atmfSrvcRegParm1 is ignored.

-- ATM Name Server (ANS)
atmfSrvcRegAns         OBJECT IDENTIFIER ::= { atmfSrvcRegTypes 2 }
-- When atmfSrvcRegServiceID has a value of atmfSrvcRegAns,
-- the value of atmfSrvcRegParm1 is ignored.


--          The Service Registry Table
--
-- The Service Registry Table is implemented by the network-side IME

atmfSrvcRegTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtmfSrvcRegEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "The table implemented by the IME on the network side of
                the ATM UNI port contains all of the services that are
                available to the user-side IME indexed by service
                identifier."
        ::= { atmfSrvcRegistryGroup 1 }
```

```
atmfSrvcRegEntry OBJECT-TYPE
        SYNTAX  AtmfSrvcRegEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "Information about a single service provider that is
                available to the user-side IME."
        INDEX   { atmfSrvcRegPort, atmfSrvcRegServiceID,
                  atmfSrvcRegAddressIndex }
        ::= { atmfSrvcRegTable 1 }

AtmfSrvcRegEntry ::=
        SEQUENCE {
                atmfSrvcRegPort
                        INTEGER,
                atmfSrvcRegServiceID
                        OBJECT IDENTIFIER,
                atmfSrvcRegATMAddress
                        AtmAddress,
                atmfSrvcRegAddressIndex
                        INTEGER,
                atmfSrvcRegParm1
                        OCTET STRING
        }

atmfSrvcRegPort OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "The value of 0 which has the special meaning of
                identifying the ATM Interface over which this message
                was received."
       ::= { atmfSrvcRegEntry 1 }

atmfSrvcRegServiceID OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "This is the service identifier which uniquely identifies
                the type of service at the address provided in the table."
        ::= { atmfSrvcRegEntry 2 }

atmfSrvcRegATMAddress OBJECT-TYPE
        SYNTAX  AtmAddress
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "This is the full address of the service.  The user-side
                IME may use this address to establish a connection
                with the service."
        ::= { atmfSrvcRegEntry 3 }

atmfSrvcRegAddressIndex OBJECT-TYPE
        SYNTAX INTEGER (1..2147483647)
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An arbitrary integer to differentiate multiple rows
                containing different ATM addresses for the same service
                on the same port."
        ::= { atmfSrvcRegEntry 4 }
```

```
atmfSrvcRegParm1 OBJECT-TYPE
        SYNTAX  OCTET STRING (SIZE (1..255))
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "An octet string whose size and meaning is determined
                by the value of atmfSrvcRegServiceID."
        ::= { atmfSrvcRegEntry 5 }
END
```

References

[AAL5] ITU-T Document TD-XVIII/10 “AAL Type 5, Draft Recommendation text for section 6 of I.363”, 29 January 1993, Geneva.

[ASN.1] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, (December, 1987).

[ASN.1-BER] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1), International Organization for Standardization. International Standard 8825, (December, 1987).

[AToMMIB] M. Ahmed, K. Tesink, "Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2", RFC 1695, 08/25/1994.

[COEXIST] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", RFC 1908, January 1996.

[IISP1.0] “Interim Interswitch Signaling Protocol, Version 1.0”, The ATM Forum Technical Committee.

[LANE1.0] "LAN Emulation Specification, Version 1.0", The ATM Forum Technical Committee.

[MIB-II] K. McCloghrie and M.T. Rose (editors), "Management Information Base for Network Management of TCP/IP-based internets; MIB-II, Request for Comments 1213, DDN Network Information Center, (March, 1991).

[PNNI1.0] "ATM Forum PNNI Specification, Version 1.0", The ATM Forum Technical Committee.

[Q.2931] ITU-T Q.2931, “B-ISDN DSS2 User-Network Interface (UNI) Layer 3 Specification for Basic Call/Connection Control”.

[SMI] M.T. Rose and K. McCloghrie, “Structure and Identification of Management Information for TCP/IP-based Internets, Request for Comments 1155”. DDN Network Information Center, (May, 1990).

[SNMP] J.D. Case, M.S. Fedor, M.L. Schoffstall, and J.R. Davin, “Simple Network Management Protocol, Request for Comments 1157”. DDN Network Information Center, (May, 1990).

[TM4.0] "ATM Forum Traffic Management Specification, Version 4.0", The ATM Forum Technical Committee.

[UNI2.0] "User-Network Interface (UNI) Specification, Version 2.0", The ATM Forum Technical Committee.

[UNI3.1] "User-Network Interface (UNI) Specification, Version 3.1", The ATM Forum Technical Committee.

[UNI4.0] "ATM User-Network Interface (UNI) Signalling Specification, Version 4.0", The ATM Forum Technical Committee.

Annex A. Network Management Access to ILMI Data

[Normative]

A.1 Introduction

This section defines an optional proxy mechanism that enables a Network Management Station (NMS) that uses SNMP to access the ATM Interface MIBs in both an ATM system to which it has direct NM access, and also in those ATM systems to which the former ATM system is indirectly connected via an ATM interface that supports ILMI.

The situation without this feature is depicted in Figure 5 where the NMS has direct access to the SNMP agent on ATM Device A, but is only able to operate on the data in the network management MIB, such as the standard RFC 1695 MIB [AToMMIB]. Neither of the ATM Interface MIBs for the ATM interface between devices A and B are accessible to the NMS.

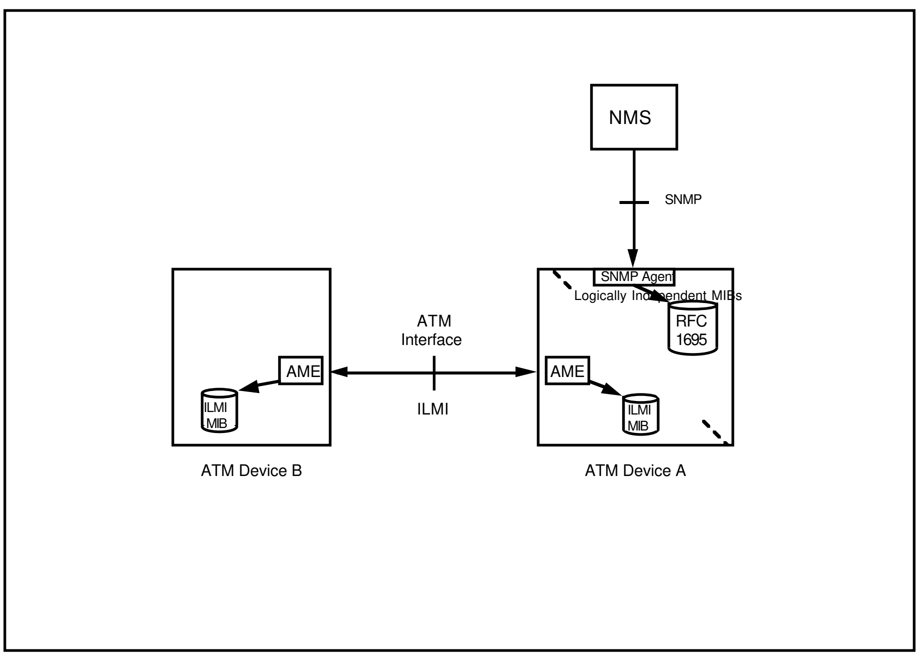


Figure 5 - ILMI MIBs not directly accessible to NMS

A.2 Overview

In addition to its role for local interface management, the data in ILMI MIBs are also useful for general Network Management functions such as configuration discovery, fault isolation and troubleshooting. For example, an ATM service provider may want to confirm the configuration of the ATM interfaces on devices attached to its network, even though it does not usually have direct NM access to those external devices.

This section defines a proxy-agent mechanism that uses the existing functions of the ILMI to provide NMS access to ATM Interface MIB data. The proxy uses the agent-role capability of the local IME to access ATM Interface MIB data in the local system, and the manager-role capability of the IME to access ATM Interface MIB data in the neighboring system. The solution defined here is depicted in Figure 6.

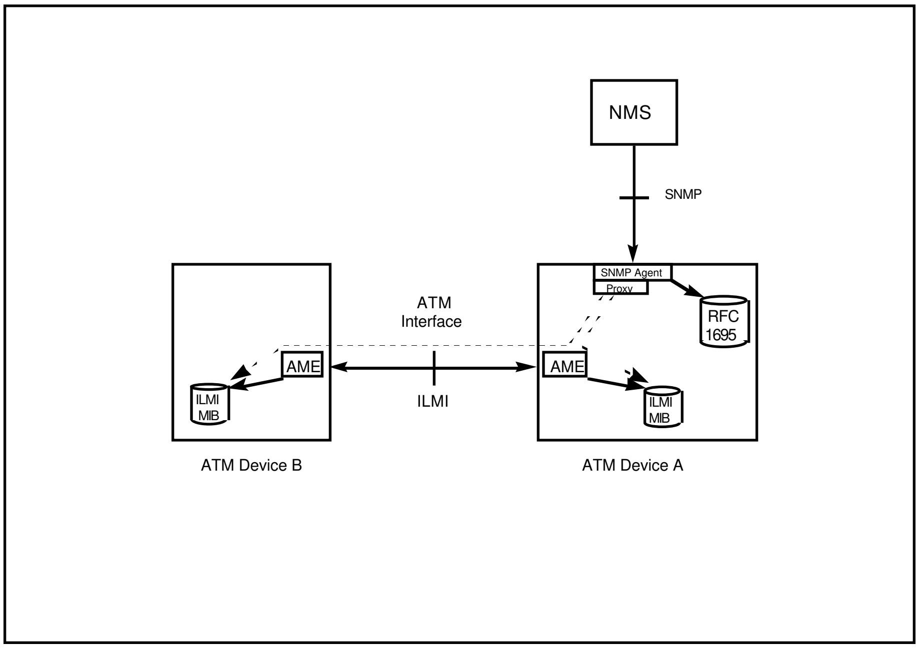


Figure 6 - NMS access to ILMI MIBs through Proxy-agent

A.3 The Proxy Approach

Since the data in ATM Interface MIBs is already accessible using SNMP via the IMEs, the simplest method to make this data accessible to external management systems is to add an SNMP proxy-agent that accepts SNMP requests from an NMS and relays them to the appropriate IME for processing as either local operations to be run against its own ATM Interface MIB or as operations to be forwarded across the ILMI interface to its peer IME for remote processing.

For each IME, two proxy-targets are defined -- one to receive requests for the local ATM Interface data, the other to handle requests for data in the neighboring ATM Interface MIB. Each proxy-target implements an independent ATM Interface MIB view including, e.g., an implementation of the system group.

When the proxy-agent receives a request from an external NMS, it must determine whether it should handle the request normally (e.g. as request for data from the RFC 1695 [AToMMIB]) or whether the request should be forwarded to one of the IME proxy-targets (and, if so, which one).

This determination is made on the basis of the community-string in the request. For each IME, two community strings will typically be defined in order to distinguish requests for the local ATM Interface MIB from requests for the remote ATM Interface MIB. How community strings are assigned to IMEs is implementation specific.

For SNMPv1 requests, the proxy-agent does not modify the PDU it forwards between the NMS and proxy-target.

SNMPv2 requests are forwarded to the IME using the SNMPv2-to-SNMPv1 proxy procedures defined in RFC 1908 [COEXIST].

When the proxy-target receives the request, it performs the SNMP operation with respect to its defined MIB view, and sends a response to the proxy-agent, which in turn sends it to the NMS.

Since the semantics of SNMP require that all variable-bindings in a single request be processed as if simultaneously, the proxy approach provides SNMP multiplexing on a per-request basis -- all variable-bindings in a single request are processed by a single proxy-target, i.e. with respect to a single ATM Interface MIB view.

If a proxy-agent receives an SNMP message containing an unknown community string, it follows the appropriate procedures as described in the RFC 1157 [SNMP].

Annex B. Support for Virtual UNIs

[Normative]

[UNI4.0] includes optional support for Virtual UNIs in a manner that is backward-compatible for user devices. VP Cross-Connects (VP Muxes) may connect multiple users to a single switch facility (i.e. interface). Each user device is assigned one or more VPCs on its own interface. The VP Mux combines all user VPCs onto the single switch facility by relaying user cells based only on their VPI; the VP Mux relays the VCI of user cells transparently (i.e. VP Muxes function as a VP switch). Explicit support is required only in switches; users devices remain unaware of the existence of the VP Mux.

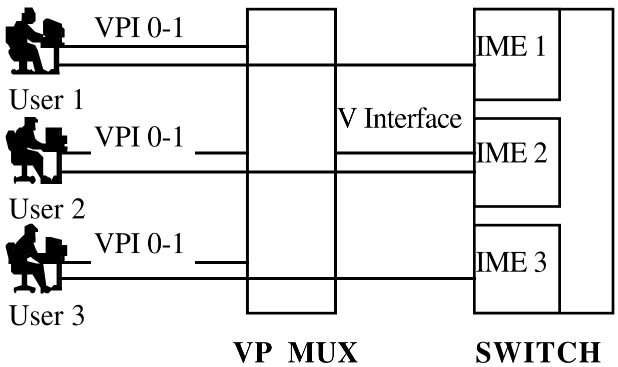
To achieve this, the VP Mux must be completely transparent to user devices. User devices continue to send and receive signalling messages on VPI=0, VCI=5, and ILMI messages use VPI=0, VCI=16.

If a switch supports VP Muxes, it is required to support multiple *virtual UNIs* on the single facility connecting it to the VP Mux. On the interface between the VP Mux and the switch, a unique VPI is used to distinguish between individual users. Switches send and receive signalling messages on VPI=X, VCI=5, and ILMI messages use VPI=X, VCI=16, where a different X is associated with each individual user.

B.1 Signalling

As shown in this example configuration, end users use VPI=VPCI=0, VCI=5 for all signalling. In order to be backwards compatible with UNI 3.1 compliant end-systems, the network should be able to associate VPIs with a unique combination of (VPCI, Signalling Channel VPI). This would allow end systems to set the VPCI equal to the VPI even when VP Mux are used in the access. The switch maintains a translation table between VPCI and VPI. The VPCI remains constant in the signalling message as it travels between the user and switch. The VPI associated with the VPCI is used at each end for the actual user traffic. The use of VPCIs is described in paragraph 5.1.2.2/Q.2931 of Chapter 2 in [UNI4.0].

As an example assume that the switch is connected to the VP Mux with an OC3 and that three users are connected to the mux with 51 Mb/s twisted pair interfaces. On the user side, each user is assigned VPI 0 and 1. These are translated to the unique values shown in the table on the switch side. Signalling and ILMI would use the default VCI on VPI zero on the user side and would be translated to VPI values 1, 3 and 5 for users 1, 2 and 3, respectively. This is shown in Figure 7below. Although this example figure shows a physically separate VP multiplexor, this functionality could be included in the end-user equipment.



END SYSTEM			NETWORK
USER	VPCI	VPI	VPI
1	0	0	1
1	1	1	2
2	0	0	3
2	1	1	4
3	0	0	5
3	1	1	6

Figure 7 - VP Mux

B.2 ILMI

Before VP Muxes were considered, each ATM Interface MIB was intended to maintain management information about a single interface. Each interface was assumed to connect a single user to a single switch port, and all ILMI information was exchanged directly between users and switches. When VP Muxes are supported, all ILMI information continues to be exchanged between users and switches, but switches must make it appear as if the VP Mux does not exist.

To do so, a switch must implement one ATM Interface MIB per *virtual UNI*. Each MIB is implemented as if it existed within the VP Mux, with the switch acting as a proxy agent. All interface indices, VPIs, and physical layer information represented in the MIB refer to those for the interface between the user and the VP Mux. Switches should consider all VPIs as VPCIs and should use the same VPI-to-VPCI translations used by signalling.

For a Virtual UNI, the interface-related parameters of the switch's IME apply to the UNI side of the VP Mux, rather than the V interface. In this sense, the switch's IME acts as a proxy for the VP Mux. Thus a switch interface that supports Virtual UNIs must keep information regarding the V interface, which is beyond that required for the multiple IMEs. For example in Figure 7above, for the number of allocated VPI bits, User 1 could have a value of 2, and User 2 could have a value of 3. Note that there may not be an IME for the V interface; this is for further study. The V interface will need to be configured (e.g. it could have 8 allocated bits), but this could be done independent of any ILMI mechanism (i.e. through the M4 interface).

For current ILMI functions, information exchange between the switch and VP Mux is unnecessary since the information is either static (e.g. interface parameters) or is known by the switch anyway (e.g. Virtual channel attributes).

B.2.1 ATM Layer Interface UNI MIB Attributes

ATM layer configuration information is relevant to both the VP Mux and the switch. The maximum number of VPCs is determined by the configuration of the VP Mux and remains static. In the example each user has a maximum of two VPCs. The maximum number of VCCs is configured at the switch, since the VP Mux must relay VCIs transparently; this information is also static. The number of configured VPCs and VCCs is really the number of VPCs and VCCs allocated to a single user that the switch is configured to process. In the above example, if the switch was configured to process 256 VCCs on VPI 1, 3, and 5, then the number of configured VCCs would be set at 256 in each user's MIB. In total the switch port would be configured to process 256 x 3 = 768 VCCs. The maximum number of VPI bits refers to the number of active VPI bits on the user to VP Mux interface. In this example one bit is active for each user. Similarly, the maximum number of VCI bits refers to the number of active VCI bits on the switch interface for each user - eight in this example.

B.2.2 Per-Virtual Path Attributes and Per-Virtual Path ABR Attributes

Each per-user ATM Interface MIB must represent Per-Virtual Path Attributes and Per-Virtual Path ABR Attributes from the VP Mux’s point of view. Switches must interpret the atmfVpcVpi and atmfVpcAbrVpi objects as VPCIs rather than VPIs.

B.2.3 Per-Virtual Channel Attributes and Per-Virtual Channel ABR Attributes

Each per-user ATM Interface MIB must represent Per-Virtual Channel Attributes and Per-Virtual Channel ABR Attributes from a point of view considering both the VP Mux’s VPI and the switch’s VCI. Switches must interpret the atmfVccVpi and atmfVccAbrVpi objects as VPCIs rather than VPIs. The atmfVccVci and atmfVccAbrVci objects are not translated.

Appendix I. ILMI FSM

[Informative]

The Finite State Machine (FSM) described in this section specifies the intended behavior for an IME on a Private User or Node.

This FSM is intended as a complement to the textual procedures described in the main body of this specification. If conflicts between the written procedures and the FSM exist, the written procedures will take precedence.

This FSM covers four potential configurations:

1. Private User attached to a Node (UNI, User-Side IME)
2. Private Node attached to a Public Node (Public UNI, User-Side IME)
3. Private Node attached to a Private User (Private UNI, Network-Side IME)
4. Private Node attached to another Private Node (Private NNI, PNNI IME)

The FSM is described in six sections:

1. An FSM Graphical View is show in section I.1.
2. All FSM States are described in section I.2.
3. All FSM Events are described in section I.3.
4. All FSM Actions are described in section I.4.
5. All SNMP Requests are shown in section I.5.
6. The FSM Summary Table is shown in section I.6.

The objects used to define the FSM (events, actions, states, transitions) are abstractions.

I.1 FSM GRAPHICAL VIEW

Figure 8 and Figure 9 show the Link Management FSM Graphical View.

Figure 10 shows the Address Registration FSM graphical view.

The FSM Graphical View is intended as a complement to the FSM Summary Table. If conflicts between the FSM Summary Table and the FSM Graphical View exist, the FSM Summary Table will take precedence.

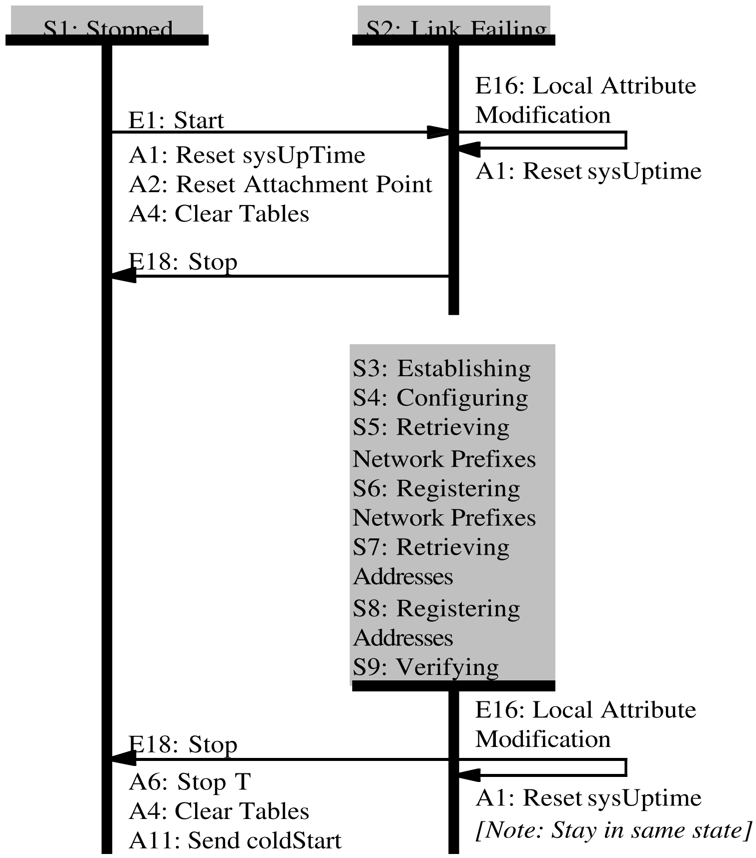


Figure 8 - Link Management FSM (Part 1)

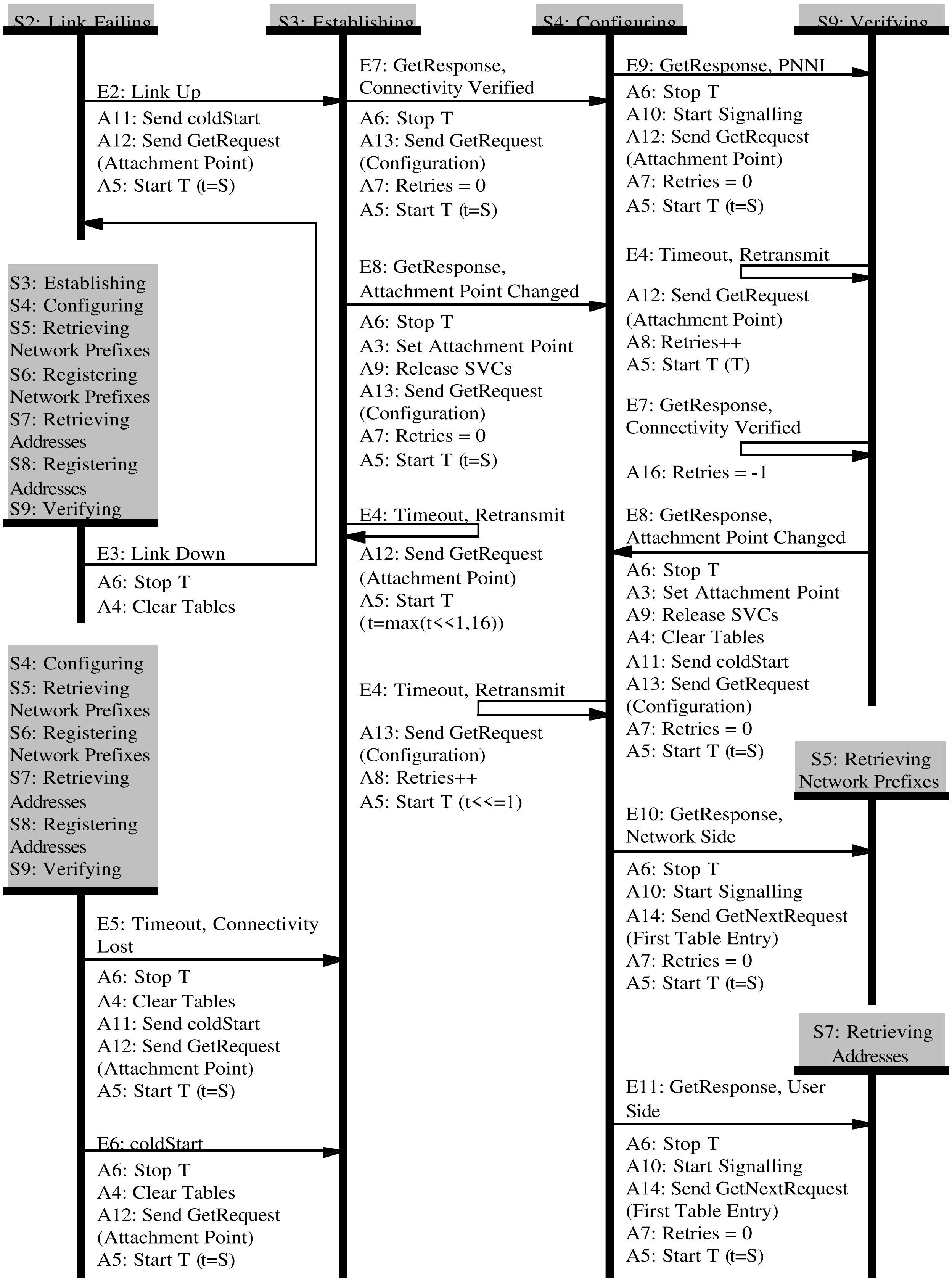


Figure 9 - Link Management FSM (Part 2)

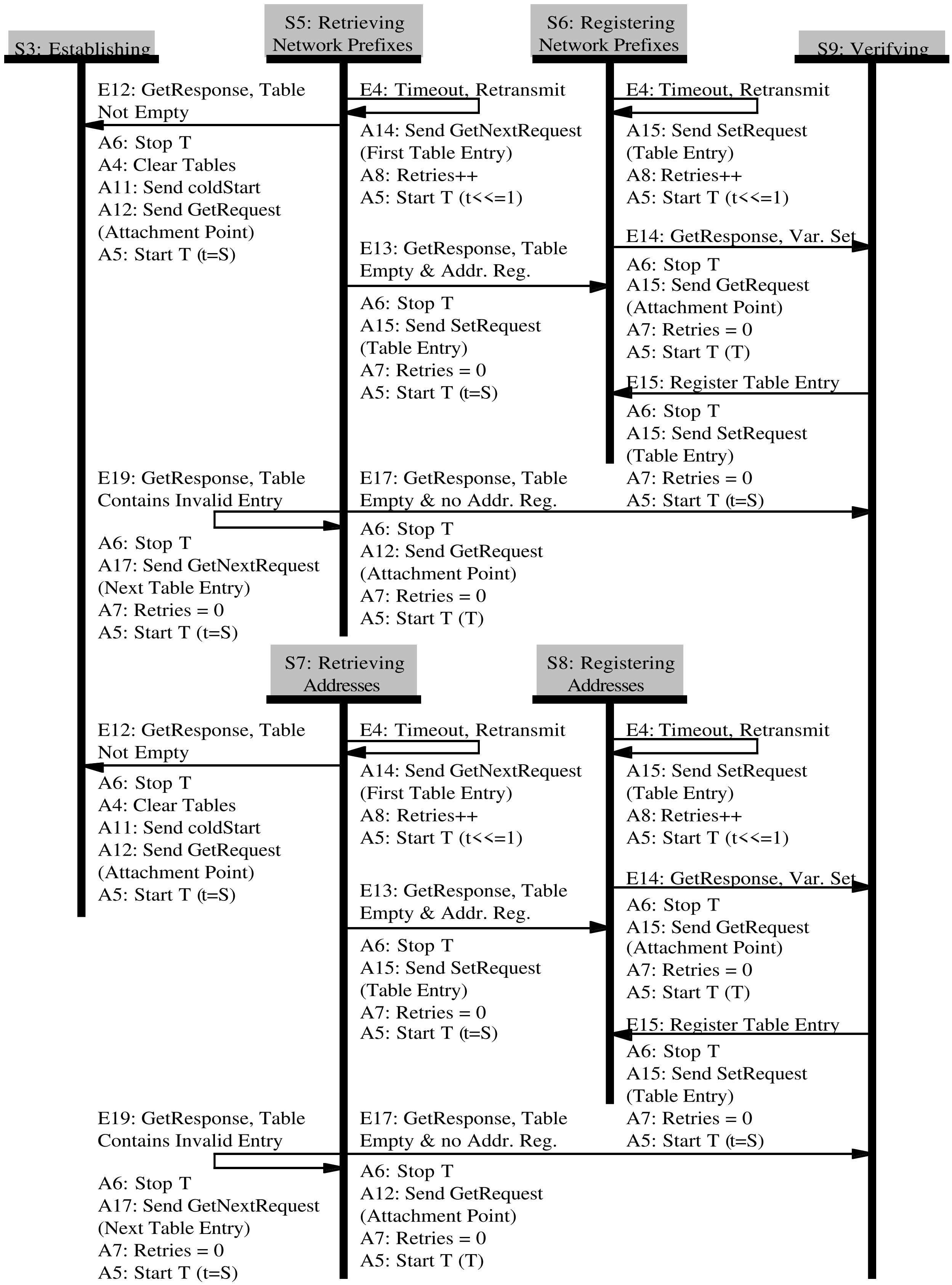

S5: Retrieving Network Prefixes
S6: Registering Network Prefixes
S3: Establishing
S9: Verifying
E12: GetResponse, Table Not Empty
A6: Stop T
A4: Clear Tables
A11: Send coldStart
A12: Send GetRequest (Attachment Point)
A5: Start T (t=S)
E4: Timeout, Retransmit
A14: Send GetNextRequest (First Table Entry)
A8: Retries++
A5: Start T (t<<=1)
E4: Timeout, Retransmit
A15: Send SetRequest (Table Entry)
A8: Retries++
A5: Start T (t<<=1)
E13: GetResponse, Table Empty & Addr. Reg.
A6: Stop T
A15: Send SetRequest (Table Entry)
A7: Retries = 0
A5: Start T (t=S)
E14: GetResponse, Var. Set
A6: Stop T
A15: Send GetRequest (Attachment Point)
A7: Retries = 0
A5: Start T (T)
E15: Register Table Entry
A6: Stop T
A15: Send SetRequest (Table Entry)
A7: Retries = 0
A5: Start T (t=S)
E19: GetResponse, Table Contains Invalid Entry
A6: Stop T
A17: Send GetNextRequest (Next Table Entry)
A7: Retries = 0
A5: Start T (t=S)
E17: GetResponse, Table Empty & no Addr. Reg.
A6: Stop T
A12: Send GetRequest (Attachment Point)
A7: Retries = 0
A5: Start T (T)
S7: Retrieving Addresses
S8: Registering Addresses
E12: GetResponse, Table Not Empty
A6: Stop T
A4: Clear Tables
A11: Send coldStart
A12: Send GetRequest (Attachment Point)
A5: Start T (t=S)
E4: Timeout, Retransmit
A14: Send GetNextRequest (First Table Entry)
A8: Retries++
A5: Start T (t<<=1)
E4: Timeout, Retransmit
A15: Send SetRequest (Table Entry)
A8: Retries++
A5: Start T (t<<=1)
E13: GetResponse, Table Empty & Addr. Reg.
A6: Stop T
A15: Send SetRequest (Table Entry)
A7: Retries = 0
A5: Start T (t=S)
E14: GetResponse, Var. Set
A6: Stop T
A15: Send GetRequest (Attachment Point)
A7: Retries = 0
A5: Start T (T)
E15: Register Table Entry
A6: Stop T
A15: Send SetRequest (Table Entry)
A7: Retries = 0
A5: Start T (t=S)
E19: GetResponse, Table Contains Invalid Entry
A6: Stop T
A17: Send GetNextRequest (Next Table Entry)
A7: Retries = 0
A5: Start T (t=S)
E17: GetResponse, Table Empty & no Addr. Reg.
A6: Stop T
A12: Send GetRequest (Attachment Point)
A7: Retries = 0
A5: Start T (T)

Figure 10 - Address Registration FSM

I.2 FSM States

Table 5 - States

Number	Name	SNMP Request(s) Outstanding	Description
S1	Stopped	None	System stopped or just booted • This is always the first state and last state • System has not yet been initialized • Waiting to start operation
S2	Link Failing	None	Link Connectivity is lost • Physical or logical link is indicating a failure • Attempting to establish Link Connectivity
S3	Establishing	GetRequest (Attachment Point)	Link Connectivity is established, but ILMI Connectivity is lost • Physical or logical link is indicating that it has reconnected, or may have received a coldStart indicating peer has restarted • ILMI Connectivity has not yet been established or has been lost • Attempting to establish ILMI Connectivity
S4	Configuring	GetRequest (Configuration)	ILMI Connectivity is established • ILMI Connectivity has been established • Configuration information has not yet been retrieved • Attempting to retrieve configuration information
S5	Retrieving Network Prefixes	GetRequest (First Network Prefix Table Entry)	Configured as Network-Side IME • Have configured as an Network-Side IME • Have not yet verified that the Network Prefix Table is empty • Attempting to verify that the Network Prefix Table is empty
S6	Registering Network Prefixes	SetRequest (Network Prefix Table Entry)	Registering Network Prefixes • Have verified that the Network Prefix Table started empty • All Network Prefixes have not yet been registered • Attempting to register Network Prefixes
S7	Retrieving Addresses	GetRequest (First Address Table Entry)	Configured as User-Side IME • Have configured as an User-Side IME • Have not yet verified that the Address Table is empty • Attempting to verify that the Address Table is empty
S8	Registering Addresses	SetRequest (Address Table Entry)	Registering Addresses • Have verified that the Address Table started empty • All Addresses have not yet been registered • Attempting to register Addresses
S9	Verifying	None, or GetRequest (Attachment Point)	Verifying ILMI Connectivity • Have configured as a PNNI IME, or as a Network-Side IME and Network Prefix registration has completed, or as a User -Side IME and Address registration has completed

			• Periodically testing ILMI Connectivity • Attempting to verify ILMI Connectivity, or waiting for next keep-alive timeout

I.3 FSM EVENTS

Table 6 - Events

Number	Name	Description
E1	Start	The system has been started
E2	Link Up	Link Connectivity has been established with the peer IME • Physical or logical link is indicating that it has reconnected
E3	Link Down	Link Connectivity has been lost with the peer IME • Physical or logical link is indicating a failure (e.g. via carrier loss, VP OAM procedures, etc.)
E4	Timeout, Retransmit	The last request should be retransmitted • The timer T has expired, and • Have not received a noError response corresponding to the last request transmitted, and • The maximum number of retries K has not been reached
E5	Timeout, Connectivity Lost	ILMI Connectivity has been lost with the peer IME • The timer T has expired, and • Have not received a noError response corresponding to the last request transmitted, and • The maximum number of retries K has been reached
E6	coldStart	The peer IME has restarted • A coldStart Trap has been received from the peer IME
E7	GetResponse, Connectivity Verified	ILMI Connectivity has been re-established or verified with the peer IME • Received a GetResponse corresponding to the last request transmitted, and • The Attachment Point information is still the same
E8	GetResponse, Attachment Point Changed	The attachment point has changed • Received a GetResponse corresponding to the last request transmitted, and • The Attachment Point information has changed
E9	GetResponse, PNNI	Configured as a PNNI IME • Received a GetResponse indicating a PNNI IME configuration
E10	GetResponse, Network Side	Configured as a Network-Side IME • Received a GetResponse indicating a Network-Side IME configuration
E11	GetResponse, User Side	Configured as a User-Side IME • Received a GetResponse indicating a User-Side IME configuration
E12	GetResponse, Table Not Empty	The table is not empty in the peer IME • Received a GetResponse corresponding to the last request transmitted, and • The error-status was noError, and • The returned variable was a table entry Status variable with a valid(1) value
E13	GetResponse, Table Empty and Address Registration	The table is empty in the peer IME and Address Registration is supported • Address Registration is supported and received a GetResponse corresponding to the last request transmitted, and • The error-status was noSuchName, or

	Supported	• The error-status was noError, and • The returned variable was not a table entry Status
E14	GetResponse, Variable Set	The last SetRequest has been acknowledged by the peer IME • Received a GetResponse corresponding to the last request transmitted, and • The error-status was noError (the variable has been set), or • The error-status was badValue (the specific value is not supported), or • The error-status was noSuchName (the variable is not supported)
E15	Register Table Entry	A table entry must be added or removed • A configuration change requires that a table entry be added or removed
E16	Local Attribute Modification	The IME must be re-initialized • Local management has modified a local ATM Layer Interface Attribute
E17	GetResponse, Table Empty and Address Registration not Supported	The table is empty in the peer IME and Address Registration is not supported • Address Registration is not supported and received a GetResponse corresponding to the last request transmitted, and • The error-status was noSuchName, or • The error-status was noError, and • The returned variable was not a table entry Status
E18	Stop	The system has been stopped • Local management has disabled ILMI
E19	GetResponse, Table Contains Invalid Entry	The table in the peer IME contains an entry with Status=Invalid • Received a GetResponse corresponding to the last request transmitted, and • The error-status was noError, and • The returned variable was a table entry Status variable with a invalid(2) value

I.4 FSM ACTIONS

Table 7 - Actions

Number	Name	Description
A1	Reset sysUpTime	Reset the local sysUpTime variable to zero
A2	Reset Attachment Point	Initialize the Attachment Point with an illegal value
A3	Set Attachment Point	Replace the previous attachment point with the current one
A4	Clear Tables	Clear all entries in the Address and/or Network Prefix Table for this IME • All dynamic address information should be cleared
A5	Start T	Start the timer T with a specified value
A6	Stop T	Stop timer T
A7	Retries = 0	Reset the number of retries to zero
A8	Retries++	Increment the number of retries
A9	Release SVCs	Clear all SVCs • ILMI has detected that it has connected with a different AME or that the peer AME has been reinitialized. Signalling and routing must be halted until auto-configuration has completed.

A10	Start Signalling	Start Signalling with the new configuration • Configure signalling as User-Side, Network-Side, or PNNI
A11	Send coldStart	Send a coldStart Trap to the peer IME
A12	Send GetRequest (Attachment Point)	Get the Attachment Point from the peer IME • Transmit a GetRequest for the Attachment Point Identification objects
A13	Send GetRequest (Configuration)	Get the peer IME's Configuration objects • Transmit a GetRequest or GetNextRequest for the Configuration objects • In order to provide efficient variables retrieval, it is recommended to issue only one Get (or GetNext) Request. If backward compatibility is required, request a limited number of parameters (such as only the UNI type). In this case, send multiple Gequests to retrieve more parameters or use default values
A14	Send GetNextRequest (First Table Entry)	Get the first Table entry Status object from the peer IME • Transmit a GetNextRequest for the Table object
A15	Send SetRequest (Table Entry)	Set a Table entry in the peer IME • Transmit a SetRequest for the Table object • Issue set prefix, simple or multiple var binds generally a switch can put as many prefixes as it will fit in 484 bytes of SNMP packet. However, due to the high number of non fully standard compliant implementations on the market, it is expected that for quite some time the switch will limit itself sending one prefix per packet. This might be a configurable feature
A16	Retries = -1	Set the number of retries to -1
A17	Send GetNextRequest (Next Table Entry)	Get the next Table entry Status object from the peer IME • Transmit a GetNextRequest for the Table object

I.5 SNMP Requests

Table 8 - SNMP Requests

Name	Objects
GetRequest (Attachment Point)	sysUpTime atmfMySystemIdentifier atmfPortMyIfIndex
GetRequest (Configuration)	atmfAtmLayerMaxVpiBits atmfAtmLayerMaxVciBits atmfAtmLayerMaxVPCs atmfAtmLayerMaxVCCs atmfAtmLayerMaxSvpcVpi atmfAtmLayerMaxSvccVpi atmfAtmLayerMinSvccVci atmfAtmLayerUniType atmfAtmLayerDeviceType atmfAtmLayerUniVersion atmfAtmLayerNniSigVersion atmfAddressRegistrationAdminStatus

	atmfVccServiceCategory (VPI=0, VCI=5) atmfVccBestEffortIndicator (VPI=0, VCI=5) atmfVccReceiveTrafficDescriptorType (VPI=0, VCI=5) atmfVccReceiveTrafficDescriptorParam1 (VPI=0, VCI=5) atmfVccReceiveTrafficDescriptorParam2 (VPI=0, VCI=5) atmfVccReceiveTrafficDescriptorParam3 (VPI=0, VCI=5) atmfVccReceiveTrafficDescriptorParam4 (VPI=0, VCI=5) atmfVccReceiveTrafficDescriptorParam5 (VPI=0, VCI=5)
GetRequest (First Network Prefix Table Entry)	atmfNetPrefixTable
GetRequest (Next Network Prefix Table Entry)	atmfNetPrefixTable
SetRequest (Network Prefix Table Entry)	atmfNetPrefixStatus
GetRequest (First Address Table Entry)	atmfAddressTable
GetRequest (Next Address Table Entry)	atmfAddressTable
SetRequest (Address Table Entry)	atmfAddressStatus

I.6 FSM SUMMARY TABLE

Table 9 - Summary

States x Events:	**S1 Stopped**	**S2 Link Failing**	**S3 Establishing**	**S4 Configuring**
E1: Start	A1: Reset sysUpTime A2: Reset Attachment Point A4: Clear Tables S2: Link Failing	N/A	N/A	N/A
E2: Link Up	N/A	A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	N/A	N/A
E3: Link Down	N/A	N/A	A6: Stop T A4: Clear Tables S2: Link Failing	A6: Stop T A4: Clear Tables S2: Link Failing
E4: Timeout, Retransmit	N/A	N/A	A12: Send GetRequest (Attachment Point)	A13: Send GetRequest (Configuration)

			A5: Start T (t=max(t<<1, 16)) S3: Establishing	A8: Retries++ A5: Start T (t <<=1) S4: Configuring
E5: Timeout, Connectivity Lost	N/A	N/A	N/A	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing
E6: coldStart	N/A	N/A	N/A	A6: Stop T A4: Clear Tables A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing
E7: GetResponse, Connectivity Verified	N/A	N/A	A6: Stop T A13: Send GetRequest (Configuration) A7: Retries = 0 A5: Start T (t=S) S4: Configuring	N/A
E8: GetResponse, Attachment Point Changed	N/A	N/A	A6: Stop T A3: Set Attachment Point A9: Release SVCs A13: Send GetRequest (Configuration) A7: Retries = 0 A5: Start T (t=S) S4: Configuring	N/A

E9: GetResponse PNNI	N/A	N/A	N/A	A6: Stop T A10: Start Signalling A12: Send GetRequest (Attachment Point) A7: Retries = 0 A5: Start T (T) S9: Verifying
E10: GetResponse, Network Side	N/A	N/A	N/A	A6: Stop T A10: Start Signalling A14: Send GetNextRequest (First Table Entry) A7: Retries = 0 A5: Start T (t=S) S5: Retrieving Network Prefixes
E11: GetResponse, User Side	N/A	N/A	N/A	A6: Stop T A10: Start Signalling A14: Send GetNextRequest (First Table Entry) A7: Retries = 0 A5: Start T (t=S) S7: Retrieving Addresses
E16: Local Attribute Modification	N/A	A1: Reset sysUpTime S2: Link Failing	A1: Reset sysUpTime S3: Establishing	A1: Reset sysUpTime S4: Configuring
E18: Stop	N/A	S1: Stopped	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped

Table 10 - Summary

States x Events:	**S 5 Retrieving Network Prefixes**	**S 6 Registering Network Prefixes**	**S 7 Retrieving Addresses**	**S 8 Registering Addresses**
E3: Link Down	A6: Stop T A4: Clear Tables S2: Link Failing	A6: Stop T A4: Clear Tables S2: Link Failing	A6: Stop T A4: Clear Tables S2: Link Failing	A6: Stop T A4: Clear Tables S2: Link Failing
E4: Timeout, Retransmit	A14: Send GetNextRequest (First Table Entry) A8: Retries++ A5: Start T (t <<=1) S5: Retrieving Network Prefixes	A15: Send SetRequest (Table Entry) A8: Retries++ A5: Start T (t <<=1) S6: Registering Network Prefixes	A14: Send GetNextRequest (First Table Entry) A8: Retries++ A5: Start T (t <<=1) S7: Retrieving Addresses	A15: Send SetRequest (Table Entry) A8: Retries++ A5: Start T (t <<=1) S8: Registering Addresses
E5: Timeout, Connectivity Lost	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing
E6: coldStart	A6: Stop T A4: Clear Tables A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	A6: Stop T A4: Clear Tables A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	A6: Stop T A4: Clear Tables A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	A6: Stop T A4: Clear Tables A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing
E12: GetResponse, Table Not Empty	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	N/A	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing	N/A

E13: GetResponse, Table Empty and Address Registration Supported	A6: Stop T A15: Send SetRequest (Table Entry) A7: Retries = 0 A5: Start T (t=S) S6: Registering Network Prefixes	N/A	A6: Stop T A15: Send SetRequest (Table Entry) A7: Retries = 0 A5: Start T (t=S) S8: Registering Addresses	N/A
E14: GetResponse, Variable Set	N/A	A6: Stop T A12: Send GetRequest (Attachment Point) A7: Retries = 0 A5: Start T (T) S9: Verifying	N/A	A6: Stop T A12: Send GetRequest (Attachment Point) A7: Retries = 0 A5: Start T (T) S9: Verifying
E16: Local Attribute Modification	A1: Reset sysUpTime S5: Retrieving Network Prefixes	A1: Reset sysUpTime S6: Registering Network Prefixes	A1: Reset sysUpTime S7: Retrieving Addresses	A1: Reset sysUpTime S8: Registering Addresses
E17: GetResponse, Table Empty and Address Registration not Supported	A6: Stop T A12: Send GetRequest (Attachment Point) A7: Retries = 0 A5: Start T (T) S9: Verifying	N/A	A6: Stop T A12: Send GetRequest (Attachment Point) A7: Retries = 0 A5: Start T (T) S9: Verifying	N/A
E18: Stop	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped
E19: GetResponse, Table Contains Invalid Entry	A6: Stop T A17: Send GetNextRequest (Next Table Entry) A7: Retries = 0 A5: Start T (t=S) S5: Retrieving Network Prefixes	N/A	A6: Stop T A17: Send GetNextRequest (Next Table Entry) A7: Retries = 0 A5: Start T (t=S) S5: Retrieving Network Prefixes	N/A

Table 11 - Summary

States x Events:	**S9 Verifying**
E3: Link Down	A6: Stop T A4: Clear Tables S2: Link Failing
E4: Timeout, Retransmit	A12: Send GetRequest (Attachment Point) A8: Retries++ A5: Start T (T) S9: Verifying
E5: Timeout, Connectivity Lost	A6: Stop T A4: Clear Tables A11: Send coldStart A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing
E6: coldStart	A6: Stop T A4: Clear Tables A12: Send GetRequest (Attachment Point) A5: Start T (t=S) S3: Establishing
E7: GetResponse, Connectivity Verified	A16: Retries = -1 S9: Verifying

E8: GetResponse, Attachment Point Changed	A6: Stop T A3: Set Attachment Point A9: Release SVCs A4: Clear Tables A11: Send coldStart A13: Send GetRequest (Configuration) A7: Retries = 0 A5: Start T (t=S) S4: Configuring
E15: Register Table Entry	A6: Stop T A15: Send SetRequest (Table Entry) A7: Retries = 0 A5: Start T (t=S) S6: Registering Network Prefixes, or S8: Registering Addresses
E16: Local Attribute Modification	A1: Reset sysUpTime S3: Establishing
E18: Stop	A6: Stop T A4: Clear Tables A11: Send coldStart S1: Stopped