Network Working Group
Request for Comments: 2515
Obsoletes: 1695
Category: Standards Track

K. Tesink, Editor
Bell Communications Research

February 1999

Definitions of Managed Objects for ATM Management

Status of this Memo

Copyright Notice

Table of Contents

1. Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects used for managing ATM-based interfaces, devices, networks and services.

This memo replaces RFC 1695 [24]. Changes relative to RFC 1695 are summarized in the MIB module's REVISION clause.

Textual Conventions used in this MIB are defined in [6] and [19].

2. The SNMP Network Management Framework

The SNMP Management Framework presently consists of five major components:

0 An overall architecture, described in RFC 2271 [1].

0 Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and described in STD 16, RFC 1155 [2], STD 16, RFC 1212 [3] and RFC 1215 [4]. The second version, called SMIv2, is described in RFC 1902 [5], RFC 1903 [6] and RFC 1904 [7].

0 Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15, RFC 1157 [8]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [9] and RFC 1906 [10].

The third version of the message protocol is called SNMPv3 and described in RFC 1906 [10], RFC 2272 [11] and RFC 2274 [12].

0 Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [8]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [13].

0 A set of fundamental applications described in RFC 2273 [14] and the view-based access control mechanism described in RFC 2275 [15].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This memo specifies a MIB module that is compliant to the SMIv2. A MIB conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (e.g., use of Counter64). Some machine readable information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

3. ATM Terminology

Some basic ATM terminologies are described in this section to facilitate defining the ATM managed objects.

3.1. VCL/VPL and VCC/VPC

There are two distinct types of ATM virtual connections: Virtual Channel Connections (VCCs) and Virtual Path Connection (VPCs). As shown in Figures 1 and 2, ATM virtual connections consist of concatenated series of virtual links which forms a path between two end points, with each concatenation occurring at an ATM switch. Virtual links of VCCs are called Virtual Channel Links (VCLs). Virtual links of VPCs are called Virtual Path Links (VPLs). The VCI and VPI fields in the ATM cell header associate each cell of a VCC with a particular VCL over a given physical link. The VPI field in the ATM cell header associates each cell of a VPC with a particular VPL over a given physical link. Switches route cells between VCLs (or VPLs) via a cross-connect function according to the cells' VCI/VPI (or VPI) values.

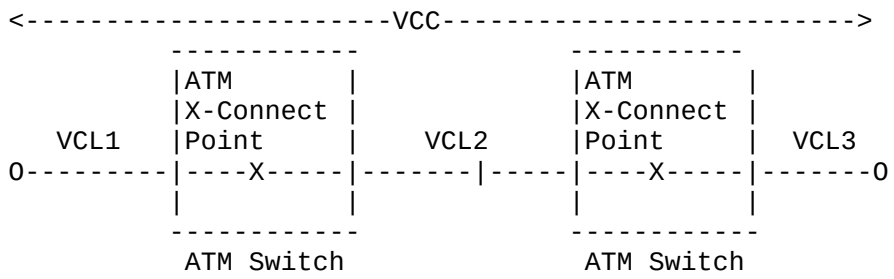

Figure 1: Virtual Channel Links and
Virtual Channel Connection

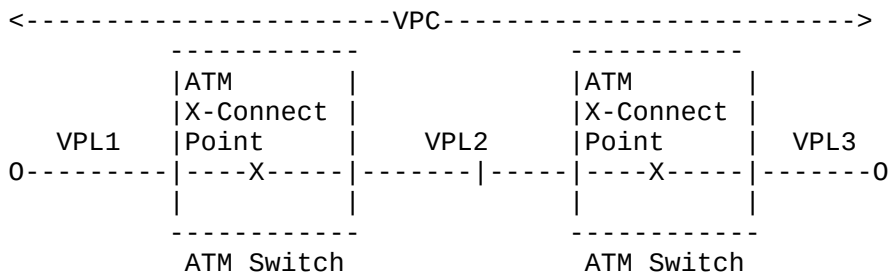

Figure 2: Virtual Path Links and
Virtual Path Connection

A single ATM end-system or switch does not support the whole end-to-end span of a VCC (or VPC). Rather, multiple ATM end-systems and/or switches each support one piece of the VCC (or VPC). That is, each ATM end-system (or ATM switch) at one end of the VCC/VPC supports its end of the VCC/VPC plus the VCL or VPL on its external interface, and each switch through which the VCC/VPC passes supports the pair of VCLs/VPLs on its external interfaces as well as the cross-connection of those VCLs/VPLs. Thus, the end-to-end management of a VCC or VPC is achieved only by appropriate management of its individual pieces in combination.

Note that for management purposes, an ATM network may be viewed as a large distributed switch by hiding all the network's internal connectivity as being internal to the distributed switch (as shown in Figure 2a). This model may for example be used for Customer Network Management (CNM) purposes.

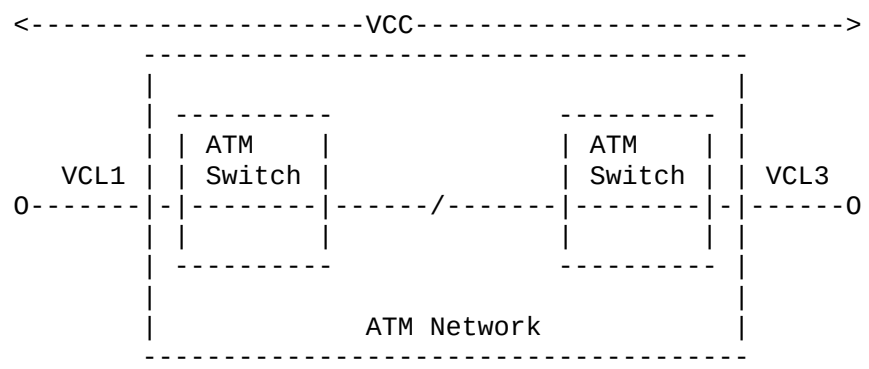

Figure 2a: ATM Network modeled as a large distributed switch

A VCC has a set of traffic characteristics (i.e., bandwidth parameters, service category parameters, etc.). VCLs inherit their traffic characteristics from the VCC of which they are a part. VCCs are bi-directional by definition. However, the traffic parameters in the two directions of a connection can be symmetric or asymmetric, i.e., the two directions can have the same or different traffic flows. A uni-directional traffic flow across a VCC is achieved by assigning a zero bandwidth in one direction. Note that in addition to the bandwidth required by the user traffic flow, bandwidth is also required for OAM cell flows, even for the zero-bandwidth direction of a uni-directional connection. These same principles apply to VPCs.

3.2. PVC, SVC and Soft PVC

A Permanent Virtual Connection (PVC) is a provisioned VCC or VPC. A Switched Virtual Connection (SVC) is a switched VCC or VPC that is set up in real-time via call set-up signaling procedures. A PVC (or an SVC) can be a point-to-point, point-to-multipoint, or multipoint-to-multipoint VCC or VPC. A Soft PVC is a connection of which portions are switched, while other portions are permanent (see Figure 3 and [22]).

Figure 3: An example of a Soft PVC

3.3. Traffic Management Parameters

3.3.1. Traffic Policing and Traffic Shaping Parameters

In order to allocate resources fairly among different users, some networks police traffic at resource access points. The traffic enforcement or policing taken at a UNI is called Usage Parameter Control (UPC) and is conceptually activated on an incoming VCL or VPL as shown in Figure 4. The use of the traffic enforcer at the ingress of the connection is to make sure that the user traffic does not exceed the negotiated traffic parameters such as the peak cell rate associated with a specific traffic descriptor type.

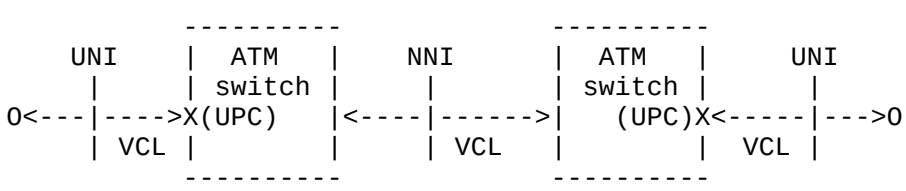

Figure 4: An Example of a UPC

In addition, traffic shaping may be performed on an outgoing VPL or VCL at a given ATM interface. The function of the ATM traffic shaper, conceptually either at the source or an egress point of the connection, is to smooth the outgoing cell traffic inter-arrival time. If policing or shaping is not performed then the policing or shaping algorithm is not activated.

3.3.2. Cell Loss Priority

To prioritize traffic during resource congestion, ATM cells are assigned one of the two types of Cell Loss Priority (CLP), CLP=0 and CLP=1. ATM cells with CLP=0 have a higher priority in regard to cell loss than ATM cells with CLP=1. Therefore, during resource congestions, CLP=1 cells are dropped before any CLP=0 cell is dropped.

3.3.3. QoS Class

RFC1695 specified that one of a number of Quality of Service (QoS) classes is assigned to a VCC or VPC by associating the object atmTrafficQoSClass with each VCL or VPL. However, new insights in ATM traffic management have caused this object to be deprecated.

3.3.4. Service Category

Replacing QoS Class, VPLs and VCLs are qualified in terms of their service category (atmServiceCategory). When properly configured, VCLs (or VPLs) concatenated to form a VCC (or VPC) will all have the same service category class as that of the VCC (or VPC).

3.4. Max Active and Max Current VPI and VCI Bits

A manager may wish to configure the maximum number of VPI and VCI bits that can be used to identify VPIs and VCIs on a given ATM interface. This value can be less than or equal to the maximum number of bits supported by the interface hardware, and is referred to in the MIB as the Max Active VPI Bits and Max Active VCI Bits.

However, a manager may not be able to configure the Max Active Bits on both ends of an ATM link. For example, the manager may not be allowed write access to the peer's MIB, or there may be hardware limitations on the peer device. Therefore, the two ATM devices may use ILMI to negotiate "Max Current" VPI and VCI bits, which is the maximum number of bits that both interfaces are willing to support. This is illustrated in Figure 5. The relationship between the different parameters is illustrated in Figure 6. Note that if ILMI negotiation is not supported, then the devices have no choice but to use the configured Max Active bits, and assume that it has been configured to the same value on both ends of the link.

IF a: Max Active VPI Bits = 6 (configured)
Max Current VPI Bits = 6 (negotiated)

IF b: Max Active VPI Bits = 8 (configured)
Max Current VPI Bits = 6 (negotiated)

IF c: Max Active VPI Bits = 8 (configured)
Max Current VPI Bits = 8 (negotiated)

IF d: Max Active VPI Bits = 8 (configured)
Max Current VPI Bits = 8 (negotiated)

(between IF a and IF b, the minimum of the two configured "Max Active VPI Bits" is 6, so both interfaces set their "Max Current VPI Bits" to 6. Since IF c and IF d both are configured with "Max Active VPI Bits" of 8, they set their "Max Current VPI Bits" to 8.)

Figure 5

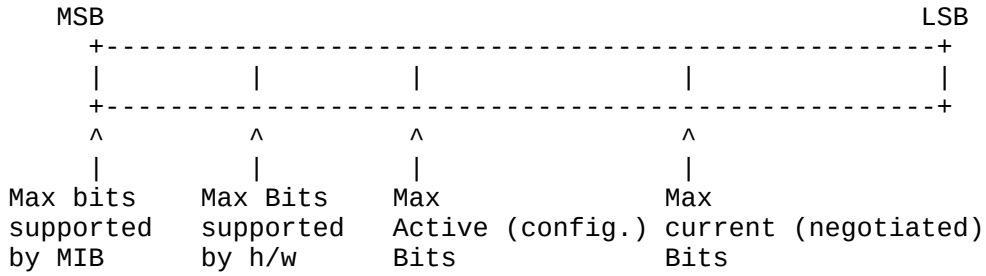

Figure 6

4. Overview

ATM management objects are used to manage ATM interfaces, ATM virtual links, ATM cross-connects, AAL5 entities and AAL5 connections supported by ATM hosts, ATM switches and ATM networks. This section provides an overview and background of how to use this MIB and other potential MIBs for this purpose.

The purpose of this memo is primarily to manage ATM PVCs. ATM SVCs are also represented by the management information in this MIB. However, full management of SVCs may require additional capabilities which are beyond the scope of this memo.

4.1. Background

In addition to the MIB module defined in this memo, other MIB modules are necessary to manage ATM interfaces, links and cross-connects. Examples include MIB II for general system and interface management [16][17], the DS3 or SONET MIBs for management of physical interfaces, and, as appropriate, MIB modules for applications that make use of ATM, such as SMDS. These MIB modules are outside the scope of this specification.

The current specification of this ATM MIB is based on SNMPv2-SMI.

4.2. Structure of the MIB

The managed ATM objects are arranged into the following tables:

(1) ATM interface configuration table
(2) ATM interface DS3 PLCP and TC sublayer tables
(3) ATM traffic parameter table
(4) ATM interface virtual link (VPL/VCL) configuration tables
(5) ATM VP/VC cross-connect tables
(6) AAL5 connection performance statistics table

Note that, managed objects for activation/deactivation of OAM cell flows and ATM traps notifying virtual connection or virtual link failures are outside the scope of this memo.

4.3. ATM Interface Configuration Table

This table contains information on ATM cell layer configuration of local ATM interfaces on an ATM device in addition to the information on such interfaces contained in the ifTable.

4.4. ATM Interface DS3 PLCP and TC Layer Tables

These tables provide performance statistics of the DS3 PLCP and TC sublayer of local ATM interfaces on a managed ATM device. DS3 PLCP and TC sublayer are currently used to carry ATM cells respectively over DS3 and SONET transmission paths.

4.5. ATM Virtual Link and Cross-Connect Tables

ATM virtual link and cross-connect tables model bi-directional ATM virtual links and ATM cross-connects. The ATM VP/VC link tables are implemented in an ATM host, ATM switch and ATM network. The ATM switch and ATM network also implement the ATM VP/VC cross-connect tables. Both link and cross-connect tables are implemented in a carrier's network for Customer Network Management (CNM) purposes.

The ATM virtual link tables are used to create, delete or modify ATM virtual links in an ATM host, ATM switch and ATM network. ATM virtual link tables along with the cross-connect tables are used to create, delete or modify ATM cross-connects in an ATM switch or ATM network (e.g., for CNM purposes).

For a PVC, the cross-connect between two VPLs is represented in the atmVpCrossConnectTable of the ATM-MIB, indexed by the atmVplCrossConnectIdentifier values for the two VPLs, and the cross-

rconnect between two VCLs is represented in the atmVcCrossConnectTable of the ATM-MIB, indexed by the atmVclCrossConnectIdentifier values for the two VCLs.

For an SVC or Soft PVC the VPL and VCL tables defined in this memo are used. Hoewever, for an SVC or Soft PVC the cross-connect between two VPLs is represented in the atmSvcVpCrossConnectTable of the ATM2-MIB, indexed by the atmVplCrossConnectIdentifier values for the two VPLs, and the cross-connect between two VCLs is represented in the atmSvcVcCrossConnectTable of the ATM2-MIB, indexed by the atmVclCrossConnectIdentifier values for the two VCLs.

Note: The ATM2-MIB module was being defined in a separate memo at the time of this publication. Please consult the RFC directory for an exact reference.

5. Application of MIB II to ATM

5.1. The System Group

For the purposes of the sysServices object in the System Group of MIB II [16], ATM is a data link layer protocol. Thus, for ATM switches and ATM networks, sysServices will have the value "2".

5.2. The Interface Group

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

This memo assumes the interpretation of the Interfaces Group to be in accordance with [17] which states that the interfaces table (ifTable) contains information on the managed resource's interfaces and that each sub-layer below the internetwork layer of a network interface is considered an interface. Thus, the ATM cell layer interface is represented as an entry in the ifTable. This entry is concerned with the ATM cell layer as a whole, and not with individual virtual connections which are managed via the ATM-specific managed objects specified in this memo. The inter-relation of entries in the ifTable is defined by Interfaces Stack Group defined in [17].

5.2.1. Support of the ATM Cell Layer by ifTable

Some specific interpretations of ifTable for the ATM cell layer follow.

Object	Use for the generic ATM layer
ifIndex	Each ATM port is represented by an ifEntry.
ifDescr	Description of the ATM interface.
ifType	The value that is allocated for ATM is 37.
ifSpeed	The total bandwidth in bits per second for use by the ATM layer.
ifPhysAddress	The interface's address at the ATM protocol sublayer; the ATM address which would be used as the value of the Called Party Address Information Element (IE) of a signalling message for a connection which either: - would terminate at this interface, or - for which the Called Party Address IE would need to be replaced by the Called Party SubAddress IE before the message was forwarded to any other interface. For an interface on which signalling is not supported, then the interface does not necessarily have an address, but if it does, then ifPhysAddress is the address which would be used as above in the event that signalling were supported. If the interface has multiple such addresses, then ifPhysAddress is its primary address. If the interface has no addresses, then ifPhysAddress is an octet string of zero length. Address encoding is as per [20]. Note that addresses assigned for purposes other than those listed above (e.g., an address associated with the service provider side of a public network UNI) may be represented through atmInterfaceSubscrAddress.
ifAdminStatus	See [17].
ifOperStatus	Assumes the value down(2) if the ATM cell layer is down.
ifLastChange	See [17].
ifInOctets	The number of received octets over the interface, i.e., the number of received, assigned cells multiplied by 53.
ifOutOctets	The number of transmitted octets over the interface, i.e., the number of transmitted, assigned cells multiplied by 53.

ifInErrors The number of cells dropped due to uncorrectable HEC errors.

ifInUnknownProtos The number of received cells discarded during cell header validation, including cells with unrecognized VPI/VCI values, and cells with invalid cell header patterns. If cells with undefined PTI values are discarded, they are also counted here.

ifOutErrors See [17].

ifName Textual name (unique on this system) of the interface or an octet string of zero length.

ifLinkUpDownTrapEnable Default is disabled (2).

ifConnectorPresent Set to false (2).

ifHighSpeed See [17].

ifHCInOctets The 64-bit version of ifInOctets; supported if required by the compliance statements in [17].

ifHCOutOctets The 64-bit version of ifOutOctets; supported if required by the compliance statements in [17].

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

6. Support of the AAL3/4 Based Interfaces

For the management of AAL3/4 CPCS layer, see [18].

7. Support of the AAL5 Managed Objects

Support of AAL5 managed objects in an ATM switch and ATM host are described below.

7.1. Managing AAL5 in a Switch

Managing AAL5 in a switch involves:

(1) performance management of an AAL5 entity as an internal resource in a switch

(2) performance management of AAL5 per virtual connection

AAL5 in a switch is modeled as shown in Figure 7 and 8. AAL5 will be managed in a switch for only those virtual connections that carry AAL5 and are terminated at the AAL5 entity in the switch. Note that, the virtual channels within the ATM UNIs carrying AAL5 will be switched by the ATM switching fabric (termed as ATM Entity in the figure) to the virtual channels on a proprietary internal interface associated with the AAL5 process (termed as AAL5 Entity in the figure). Therefore, performance management of the AAL5 resource in the switch will be modeled using the ifTable through an internal (pseudo-ATM) virtual interface and the AAL5 performance management per virtual connection will be supported using an additional AAL5 connection table in the ATM MIB. The association between the AAL5 virtual link at the proprietary virtual, internal interface and the ATM virtual link at the ATM interface will be derived from the virtual channel cross-connect table and the virtual channel link table in the ATM MIB. Note that for the proprietary virtual interface the traffic transmit and receive conventions in the virtual channel link table are as follows:

```
Transmitting traffic:  ATM Entity     --->  AAL5 Entity
Receiving traffic:     ATM Entity     <---  AAL5 Entity
```

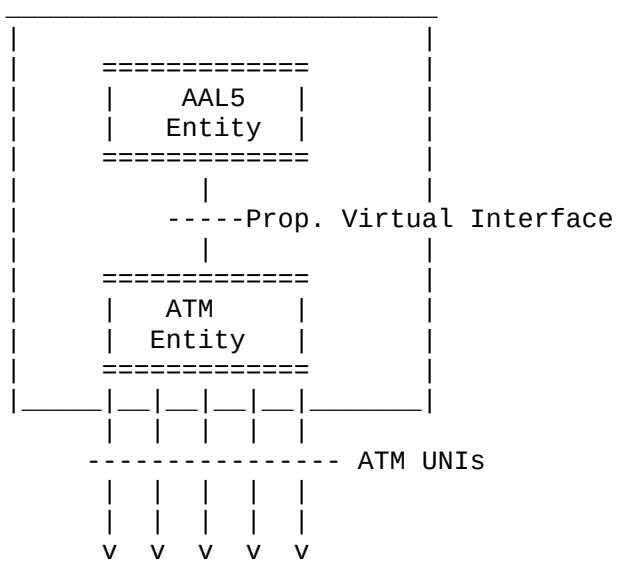

Figure 7: Model of an AAL5 Entity in a Switch

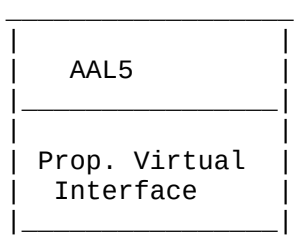

Figure 8: AAL5 Entity's Interface Stack in a Switch

7.2. Managing AAL5 in a Host

Managing AAL5 in a host involves managing the AAL5 sublayer interface as shown in Figure 9 and 10. The AAL5 sublayer is stacked directly over the ATM sublayer. The ifTable is applied to the AAL5 sublayer as defined in Section 10.3.

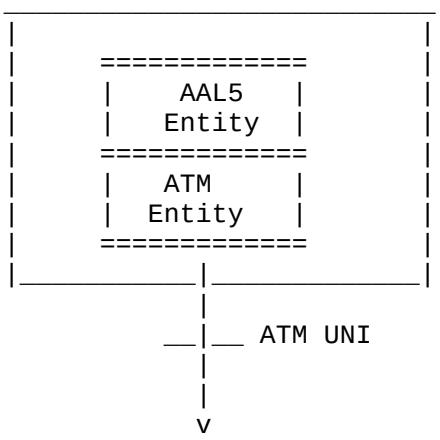

Figure 9: Model of an AAL5 Entity in a Host

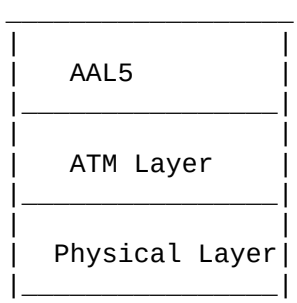

Figure 10: AAL5 Entity's Interface Stack in a Host

7.3. Support of AAL5 by ifTable

The AAL5 entity in an ATM device (e.g., switch or host) is managed using the ifTable. There are additional counters specified for AAL5 than those specified in the ATM B-ICI document [21]. Specific interpretations of ifTable for the AAL5 CPCS layer are as follows.

Object	Use for AAL5 CPCS layer entity
ifIndex	Each AAL5 entity is represented by an ifEntry.
ifDescr	Description of the AAL5 entity.
ifType	The value that is allocated for AAL5 is 49.
ifMtu	Set to the largest PDU size for the AAL5 CPCS layer that can be processed by the AAL5 entity.
ifSpeed	Set to 0.
ifPhysAddress	An octet string of zero length.
ifAdminStatus	See [17].
ifOperStatus	Assumes the value down(2) if the AAL5 layer is down.
ifLastChange	See [17].
ifInOctets	The number of received AAL5 CPCS PDU octets.
ifOutOctets	The number of AAL5 CPCS PDU octets transmitted.
ifInUcastPkts	The number of received AAL5 CPCS PDUs passed to a higher-layer.
ifOutUcastPkts	The number of AAL5 CPCS PDUs received from a higher-layer for transmission. [Note: The number of AAL5 PDUs actually transmitted is the number received from a higher-layer for transmission minus any which are counted by ifOutErrors and ifOutDiscards.]

ifInErrors Number of errored AAL5 CPCS PDUs received. The types of errors counted include CRC-32 errors, SAR time-out errors, and oversized SDU errors.

ifInUnknownProtos Set to 0.

ifInDiscards Number of received AAL5 CPCS PDUs discarded. Possible reason may be input buffer overflow.

ifOutErrors Number of AAL5 CPCS PDUs that could not be transmitted due to errors.

ifOutDiscards Number of AAL5 CPCS PDUs received for transmission that are discarded. Possible reason may be output buffer overflow.

ifInMulticastPkts Set to 0.

ifInBroadcastPkts Set to 0.

ifOutMulticastPkts Set to 0.

ifOutBroadcastPkts Set to 0.

ifName Textual name (unique on this system) of the AAL5 entity or an octet string of zero length.

ifHighSpeed Set to 0.

ifConnectorPresent Set to false (2).

ifPromiscuousMode Set to false(2).

ifLinkUpDownTrapEnable Default is disabled (2).

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

7.4. Support of Proprietary Virtual Interface by ifTable

Specific interpretations of ifTable for the proprietary virtual, internal interface associated with an AAL5 entity in an ATM switch are as follows.

Object	Use for proprietary virtual, internal interface associated with AAL entities
ifIndex	Each proprietary virtual, internal interface associated with AAL entities is represented by an ifEntry.
ifDescr	Description of the proprietary virtual, internal interface associated with AAL entities.
ifType	The value that is allocated for proprietary virtual, internal interface is 53.
ifSpeed	See [17]. Set to 0 if the speed is not known.
ifPhysAddress	See [17]. An octet string of zero length if no address is used for this interface.
ifAdminStatus	See [17].
ifOperStatus	See [17].
ifLastChange	See [17].
ifName	Textual name (unique on this system) of the interface or an octet string of zero length.
ifHighSpeed	See [17]. Set to 0 if the speed is not known.
ifConnectorPresent	Set to false (2).
ifLinkUpDownTrapEnable	Default is disabled (2).
ifAlias	The non-volatile 'alias' name for the interface as specified by a network manager.

7.5. AAL5 Connection Performance Statistics Table

An AAL5 connection table is used to provide AAL5 performance information for each AAL5 virtual connection that is terminated at the AAL5 entity contained within an ATM switch or host.

8. ILMI MIBs and the ATM Managed Objects

The ILMI MIBs are specified by the ATM Forum as a set of several MIBs, all currently defined in the ILMI Specification [23]. The ILMI protocols and MIBs allow two connected ATM Interface Management Entities (IMEs) to exchange bi-directional parameters, mainly to facilitate auto-configuration between ATM peer entities. The support of the ATM management functions by the ILMI MIBs and those contained in this memo are compared in Table 1. In this table, "yes" in the "ILMI MIBs" column indicates that the management functions are supported by the ILMI MIBs. The parenthesized numbers in the "This memo" column correspond to the sets of tables enumerated in Section 6.2.

For that subset of management information which the ILMI MIBs and this memo have in common, every effort has been made to retain identical semantics and syntax, even though the MIB objects are identified using different OBJECT IDENTIFIERs.

Table 1 - Structuring of ATM Managed Objects

ATM Mgmt.Inf.	ATM Managed Objects	This memo	ILMI MIBs
Local Interface Information:			
ATM interface: physical layer configuration	(1) port identifier	ATM MIB (1)* MIB II	yes *
	(2) physical transmission types		
	(3) operational status		
	(4) administrative status		**
	(5) last change status		
ATM interface: cell layer configuration	(1) active VPI/VCI fields	ATM MIB (1)	yes **
	(2) maximum number of VPCs/VCCs		
	(3) configured VPCs/VCCs		
	(4) ILMI VPI/VCI values		
	(5) Neighbor system info		
	(6) Max. number of VPI/VCI bits		yes
	(7) ATM Subscribed Address		
ATM interface: cell layer performance	(1) received/transmitted cells	MIB II	yes
	(2) cells with HEC error		
	(3) cell header validation errors		

ATM interface: PLCP & TC layer performance	(1)DS3 PLCP severely errored framing seconds	ATM MIB (2)		
	(2)DS3 PLCP unavailable seconds		no	
	(3)DS3 PLCP alarm state			
	(4)out of cell delineation events			
	(5)TC alarm state			
VP/VC link: configuration	(1)VPI or VPI/VCI value	ATM MIB (3,4)		
	(2)VCL or VPL operational status		yes	
	(3)VCL/VPL administrative status		***	
	(4)VCL/VPL last change status			
	(5)transmit/receive traffic/ service category parameters			
	(6)AAL type			
	(7)transmit/receive AAL5 SDU size			
	(8)AAL5 encapsulation type			
	(9)connection topology type			
	(10)use of call control			
VP/VC Cross-connect: configuration	(1)cross-connect identifier			
	(2)port identifier of one end			
	(3)port identifier of the other end	ATM MIB (5)	no	
	(4)VPI or VPI/VCI value of one end			
	(5)VPI or VPI/VCI value of the other end			
	(6)VC/VP cross-connect operational status			
	(7)VC/VP cross-connect administrative status			
	(8)VC/VP last change status			
VCC AAL5 CPCS layer: performance	(1)PDUs discarded for CRC errors	ATM MIB (6)		
	(2)PDUs discarded due to reassembly time out		no	
	(3)PDUs discarded due to large SDUs			
AAL5 entity:	(1)received/transmitted PDUs			
	(2)PDUs discarded due to protocol errors	MIB II	no	
	(3)a set of configuration/state parameters			

*The operational, administrative, and last change status of the ATM interface and the physical transmission type shall be supported by the interface table in MIB II [16][17]. ILMI does not contain the administrative and last change status of the ATM interface.

** The ILMI MIB contains read-only objects for various parameters at the ATM interface level.

***The ILMI MIBs contain local and end-to-end operational status of the VPC/VCC segment. However, it does not contain the VPC/VCC administrative and last change status and the VCC AAL information.

9. Definitions

```
ATM-MIB DEFINITIONS ::= BEGIN

IMPORTS
   MODULE-IDENTITY, OBJECT-TYPE,
   Counter32, Integer32, IpAddress, mib-2
       FROM SNMPv2-SMI
   DisplayString, RowStatus, TruthValue
       FROM SNMPv2-TC
   MODULE-COMPLIANCE, OBJECT-GROUP
       FROM SNMPv2-CONF
   InterfaceIndex, ifIndex
       FROM IF-MIB
   AtmAddr, AtmConnKind, AtmConnCastType,
   AtmServiceCategory, AtmTrafficDescrParamIndex,
   AtmVpIdentifier, AtmVcIdentifier,
   AtmVorXAdminStatus, AtmVorXLastChange,
   AtmVorXOperStatus, atmNoClpNoScr
       FROM ATM-TC-MIB;


atmMIB MODULE-IDENTITY
     LAST-UPDATED "9810191200Z"
     ORGANIZATION "IETF AToM MIB Working Group"
     CONTACT-INFO
       "          Kaj Tesink
         Postal:  Bellcore
                  331 Newman Springs Road
                  Red Bank, NJ 07701
         Tel:     732-758-5254
         Fax:     732-758-2269
         E-mail:  kaj@bellcore.com"
     DESCRIPTION
      "This is the MIB Module for ATM and AAL5-related
       objects for managing ATM interfaces, ATM virtual
```

```
          links, ATM cross-connects, AAL5 entities, and
          and AAL5 connections."
        REVISION      "9810191200Z"
        DESCRIPTION
         "The initial revision of this module was published
          as RFC 1695. Key revisions include:
          o  Textual Conventions and OBJECT IDENTITIES have
             been moved to a separate MIB module.
          o  Applicability of objects to PVCs, SVCs and Soft
             PVCs has been clarified.
          o  DEFVAL clauses have been added.
          o  The relationship of ifIndex values with different
             layers and sublayers related to ATM has been
             clarified.
          o  atmTrafficQosClass has been deprecated
             and replaced with atmServiceCategory.
          o  atmInterfaceCurrentMaxVpiBits and
             atmInterfaceCurrentMaxVciBits have been added with
             a description on their relationship with other
             objects.
          o  atmInterfaceAddressType and atmInterfaceAdminAddress
             have been deprecated and replaced by
             atmInterfaceSubscrAddress.
          o  atmInterfaceTCAlarmState has been clarified.
          o  atmTrafficDescrParamIndexNext has been introduced
             in order to provide a manager a free
             atmTrafficDescrParamIndex value.
          o  The atmTrafficFrameDiscard capability has been added.
          o  A connection topology type (atmVpl/VclCastType) and
             a call control type (atmVpl/VclConnKind) have been
             added.
          o  aal2 has been added to atmVccAalType."
        REVISION      "9406072245Z"
        DESCRIPTION
         "The RFC1695 version of this MIB module."
        ::= { mib-2 37 }


    atmMIBObjects  OBJECT IDENTIFIER ::= {atmMIB 1}

    -- {atmMIBObjects 1} has been moved to a separate
    -- specification [19].


    -- This ATM MIB Module consists of the following tables:
    -- (1) ATM Interface configuration table
    -- (2) ATM Interface DS3 PLCP table
    -- (3) ATM Interface TC Sublayer table
```

```
     -- (4) Atm Traffic Descriptor table
     -- (5) ATM Interface VPL configuration table
     -- (6) ATM Interface VCL configuration table
     -- (7) ATM VP Cross Connect table (for PVCs)
     -- (8) ATM VC Cross Connect table (for PVCs)
     -- (9) ATM Interface AAL5 VCC performance statistics
     --     table

     --    ATM Interface Configuration Parameters Table

     -- This table contains ATM specific
     -- configuration information associated with
     -- an ATM interface beyond those
     -- supported using the ifTable.



     atmInterfaceConfTable  OBJECT-TYPE
          SYNTAX         SEQUENCE OF AtmInterfaceConfEntry
          MAX-ACCESS     not-accessible
          STATUS         current
          DESCRIPTION
           "This table contains ATM local interface
            configuration parameters, one entry per ATM
            interface port."
          ::= { atmMIBObjects 2 }

     atmInterfaceConfEntry    OBJECT-TYPE
          SYNTAX         AtmInterfaceConfEntry
          MAX-ACCESS     not-accessible
          STATUS         current
          DESCRIPTION
           "This list contains ATM interface configuration
            parameters and state variables and is indexed
            by ifIndex values of ATM interfaces."
          INDEX { ifIndex }
          ::= { atmInterfaceConfTable  1}

     AtmInterfaceConfEntry    ::= SEQUENCE  {
          atmInterfaceMaxVpcs             INTEGER,
          atmInterfaceMaxVccs             INTEGER,
          atmInterfaceConfVpcs            INTEGER,
          atmInterfaceConfVccs            INTEGER,
          atmInterfaceMaxActiveVpiBits    INTEGER,
          atmInterfaceMaxActiveVciBits    INTEGER,
          atmInterfaceIlmiVpi             AtmVpIdentifier,
          atmInterfaceIlmiVci             AtmVcIdentifier,
```

```
          atmInterfaceAddressType         INTEGER,
          atmInterfaceAdminAddress        AtmAddr,
          atmInterfaceMyNeighborIpAddress IpAddress,
          atmInterfaceMyNeighborIfName    DisplayString,
          atmInterfaceCurrentMaxVpiBits   INTEGER,
          atmInterfaceCurrentMaxVciBits   INTEGER,
          atmInterfaceSubscrAddress       AtmAddr
               }


     atmInterfaceMaxVpcs OBJECT-TYPE
          SYNTAX         INTEGER  (0..4096)
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The maximum number of VPCs (PVPCs and SVPCs)
            supported at this ATM interface. At the ATM UNI,
            the maximum number of VPCs (PVPCs and SVPCs)
            ranges from 0 to 256 only."
          ::= { atmInterfaceConfEntry 1}

     atmInterfaceMaxVccs OBJECT-TYPE
          SYNTAX         INTEGER  (0..65536)
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The maximum number of VCCs (PVCCs and SVCCs)
            supported at this ATM interface."
          ::= { atmInterfaceConfEntry 2}

     atmInterfaceConfVpcs  OBJECT-TYPE
          SYNTAX         INTEGER  (0..4096)
          MAX-ACCESS     read-only
          STATUS         current
          DESCRIPTION
           "The number of VPCs (PVPC, Soft PVPC and SVPC)
            currently in use at this ATM interface.  It includes
            the number of PVPCs and Soft PVPCs that are configured
            at the interface, plus the number of SVPCs
            that are currently  established at the
            interface.

            At the ATM UNI, the configured number of
            VPCs (PVPCs and SVPCs) can range from
            0 to 256 only."
          ::= { atmInterfaceConfEntry 3}

     atmInterfaceConfVccs  OBJECT-TYPE
```

```
          SYNTAX         INTEGER  (0..65536)
          MAX-ACCESS     read-only
          STATUS         current
          DESCRIPTION
           "The number of VCCs (PVCC, Soft PVCC and SVCC)
            currently in use at this ATM interface.  It includes
            the number of PVCCs and Soft PVCCs that are configured
            at the interface, plus the number of SVCCs
            that are currently  established at the
            interface."
          ::= { atmInterfaceConfEntry 4}

     atmInterfaceMaxActiveVpiBits  OBJECT-TYPE
          SYNTAX         INTEGER (0..12)
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The  maximum number of active VPI bits
            configured for use at the ATM interface.
            At the ATM UNI, the maximum number of active
            VPI bits configured for use ranges from
            0 to 8 only."
          ::= { atmInterfaceConfEntry 5}

     atmInterfaceMaxActiveVciBits  OBJECT-TYPE
          SYNTAX         INTEGER (0..16)
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The maximum number of active VCI bits
            configured for use at this ATM interface."
          ::= { atmInterfaceConfEntry 6}

     atmInterfaceIlmiVpi OBJECT-TYPE
          SYNTAX         AtmVpIdentifier
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The VPI value of the VCC supporting
            the ILMI at this ATM interface.  If the values of
            atmInterfaceIlmiVpi and atmInterfaceIlmiVci are
            both equal to zero then the ILMI is not
            supported at this ATM interface."
          DEFVAL { 0 }
          ::= { atmInterfaceConfEntry 7}

     atmInterfaceIlmiVci OBJECT-TYPE
          SYNTAX         AtmVcIdentifier
```

```
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The VCI value of the VCC supporting
            the ILMI at this ATM interface.  If the values of
            atmInterfaceIlmiVpi and atmInterfaceIlmiVci are
            both equal to zero then the ILMI is not
            supported at this ATM interface."
          DEFVAL { 16 }
          ::= { atmInterfaceConfEntry 8}

     atmInterfaceAddressType  OBJECT-TYPE
          SYNTAX         INTEGER {
                            private(1),
                            nsapE164(2),
                            nativeE164(3),
                            other(4)
                              }
          MAX-ACCESS     read-only
          STATUS         deprecated
          DESCRIPTION
           "The type of primary ATM address configured
            for use at this ATM interface."
          ::= { atmInterfaceConfEntry 9 }

     --  The atmInterfaceAdminAddress object has been replaced by
     --  atmInterfaceSubscrAddress.

     atmInterfaceAdminAddress OBJECT-TYPE
          SYNTAX         AtmAddr
          MAX-ACCESS     read-only
          STATUS         deprecated
          DESCRIPTION
           "The primary address assigned for administrative purposes,
            for example, an address associated with the
            service provider side of a public network UNI
            (thus, the value of this address corresponds
            with the value of ifPhysAddress at the host side).
            If this interface has no assigned administrative
            address, or when the address used for
            administrative purposes is the same as that used
            for ifPhysAddress, then this is an octet string of
            zero length."
          ::= { atmInterfaceConfEntry 10 }

     atmInterfaceMyNeighborIpAddress OBJECT-TYPE
          SYNTAX         IpAddress
          MAX-ACCESS     read-write
```

```
          STATUS         current
          DESCRIPTION
           "The IP address of the neighbor system connected to
            the  far end of this interface, to which a Network
            Management Station can send SNMP messages, as IP
            datagrams sent to UDP port 161, in order to access
            network management information concerning the
            operation of that system.  Note that the value
            of this object may be obtained in different ways,
            e.g., by manual configuration, or through ILMI
            interaction with the neighbor system."
          ::= { atmInterfaceConfEntry 11 }

     atmInterfaceMyNeighborIfName OBJECT-TYPE
          SYNTAX         DisplayString
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The textual name of the interface on the neighbor
            system on the far end of this interface, and to
            which this interface connects.  If the neighbor
            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.  If this interface does not have a
            textual name, the value of this object is a zero
            length string.  Note that the value of this object
            may be obtained in different ways, e.g., by manual
            configuration, or through ILMI interaction with
            the neighbor system."
          ::= { atmInterfaceConfEntry 12 }

     atmInterfaceCurrentMaxVpiBits    OBJECT-TYPE
          SYNTAX         INTEGER  (0..12)
          MAX-ACCESS     read-only
          STATUS         current
          DESCRIPTION
           "The maximum number of VPI Bits that may
            currently be used at this ATM interface.
            The value is the minimum of
            atmInterfaceMaxActiveVpiBits, and the
            atmInterfaceMaxActiveVpiBits of the interface's
            UNI/NNI peer.

            If the interface does not negotiate with
            its peer to determine the number of VPI Bits
            that can be used on the interface, then the
```

```
            value of this object must equal
            atmInterfaceMaxActiveVpiBits."
          ::= { atmInterfaceConfEntry 13 }

     atmInterfaceCurrentMaxVciBits    OBJECT-TYPE
          SYNTAX         INTEGER  (0..16)
          MAX-ACCESS     read-only
          STATUS         current
          DESCRIPTION
           "The maximum number of VCI Bits that may
            currently be used at this ATM interface.
            The value is the minimum of
            atmInterfaceMaxActiveVciBits, and the
            atmInterfaceMaxActiveVciBits of the interface's
            UNI/NNI peer.

            If the interface does not negotiate with
            its peer to determine the number of VCI Bits
            that can be used on the interface, then the
            value of this object must equal
            atmInterfaceMaxActiveVciBits."
          ::= { atmInterfaceConfEntry 14 }

     atmInterfaceSubscrAddress OBJECT-TYPE
          SYNTAX         AtmAddr
          MAX-ACCESS     read-write
          STATUS         current
          DESCRIPTION
           "The identifier assigned by a service provider
            to the network side of a public network UNI.
            If this interface has no assigned service provider
            address, or for other interfaces this is an octet string
            of zero length."
          ::= { atmInterfaceConfEntry 15 }

     -- The ATM Interface DS3 PLCP Table

     -- This table contains the DS3 PLCP configuration and
     -- state parameters of those ATM interfaces
     -- which use DS3 PLCP for carrying ATM cells over DS3.

     atmInterfaceDs3PlcpTable  OBJECT-TYPE
          SYNTAX       SEQUENCE OF AtmInterfaceDs3PlcpEntry
          MAX-ACCESS   not-accessible
          STATUS       current
          DESCRIPTION
           "This table contains ATM interface DS3 PLCP
            parameters and state variables, one entry per
```

```
        ATM interface port."
      ::= { atmMIBObjects 3}

atmInterfaceDs3PlcpEntry OBJECT-TYPE
      SYNTAX         AtmInterfaceDs3PlcpEntry
      MAX-ACCESS     not-accessible
      STATUS         current
      DESCRIPTION
       "This list contains DS3 PLCP parameters and
        state variables at the ATM interface and is
        indexed by the ifIndex value of the ATM interface."
      INDEX    { ifIndex }
      ::= { atmInterfaceDs3PlcpTable 1}

AtmInterfaceDs3PlcpEntry    ::= SEQUENCE  {
      atmInterfaceDs3PlcpSEFSs       Counter32,
      atmInterfaceDs3PlcpAlarmState  INTEGER,
      atmInterfaceDs3PlcpUASs        Counter32
        }


atmInterfaceDs3PlcpSEFSs  OBJECT-TYPE
      SYNTAX         Counter32
      MAX-ACCESS     read-only
      STATUS         current
      DESCRIPTION
       "The number of DS3 PLCP Severely Errored Framing
        Seconds (SEFS). Each SEFS represents a
        one-second interval which contains
        one or more SEF events."
      ::= { atmInterfaceDs3PlcpEntry 1}

atmInterfaceDs3PlcpAlarmState    OBJECT-TYPE
      SYNTAX         INTEGER {
                        noAlarm(1),
                        receivedFarEndAlarm(2),
                        incomingLOF(3)
                              }
      MAX-ACCESS     read-only
      STATUS         current
      DESCRIPTION
       "This variable indicates if there is an
        alarm present for the DS3 PLCP.  The value
        receivedFarEndAlarm means that the DS3 PLCP
        has received an incoming Yellow
        Signal, the value incomingLOF means that
        the DS3 PLCP has declared a loss of frame (LOF)
        failure condition, and the value noAlarm
```

```
          means that there are no alarms present.
          Transition from the failure to the no alarm state
          occurs when no defects (e.g., LOF) are received
          for more than 10 seconds."
        ::= { atmInterfaceDs3PlcpEntry 2}

    atmInterfaceDs3PlcpUASs  OBJECT-TYPE
        SYNTAX         Counter32
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
         "The counter associated with the number of
          Unavailable Seconds encountered by the PLCP."
        ::= { atmInterfaceDs3PlcpEntry 3}


    -- The ATM Interface TC Sublayer Table

    -- This table contains TC sublayer configuration and
    -- state parameters of those ATM interfaces
    -- which use TC sublayer for carrying ATM cells over
    -- SONET/SDH or DS3.


    atmInterfaceTCTable  OBJECT-TYPE
        SYNTAX         SEQUENCE OF AtmInterfaceTCEntry
        MAX-ACCESS     not-accessible
        STATUS         current
        DESCRIPTION
         "This table contains ATM interface TC
          Sublayer parameters and state variables,
          one entry per ATM interface port."
        ::= { atmMIBObjects 4}

    atmInterfaceTCEntry OBJECT-TYPE
        SYNTAX         AtmInterfaceTCEntry
        MAX-ACCESS     not-accessible
        STATUS         current
        DESCRIPTION
         "This list contains TC Sublayer parameters
          and state variables at the ATM interface and is
          indexed by the ifIndex value of the ATM interface."
        INDEX  {ifIndex }
        ::= { atmInterfaceTCTable 1}

    AtmInterfaceTCEntry    ::= SEQUENCE  {
        atmInterfaceOCDEvents      Counter32,
        atmInterfaceTCAlarmState   INTEGER
```

```
        }

    atmInterfaceOCDEvents  OBJECT-TYPE
         SYNTAX         Counter32
         MAX-ACCESS     read-only
         STATUS         current
         DESCRIPTION
          "The number of times the Out of Cell
           Delineation (OCD) events occur.  If seven
           consecutive ATM cells have Header Error
           Control (HEC) violations, an OCD event occurs.
           A high number of OCD events may indicate a
           problem with the TC Sublayer."
         ::= { atmInterfaceTCEntry 1}


    atmInterfaceTCAlarmState    OBJECT-TYPE
         SYNTAX         INTEGER {
                           noAlarm(1),
                           lcdFailure(2)
                                 }
         MAX-ACCESS     read-only
         STATUS         current
         DESCRIPTION
          "This variable indicates if there is an
           alarm present for the TC Sublayer.  The value
           lcdFailure(2) indicates that the TC Sublayer
           is currently in the Loss of Cell Delineation
           (LCD) defect maintenance state.  The value
           noAlarm(1) indicates that the TC Sublayer
           is currently not in the LCD defect
           maintenance state."
          ::= { atmInterfaceTCEntry 2}

    --  ATM Traffic Descriptor Parameter Table

    -- This table contains a set of self-consistent
    -- ATM traffic parameters including the
    -- ATM traffic service category.

    -- The ATM virtual link tables (i.e., VPL and VCL tables)
    -- will use this ATM Traffic Descriptor table
    -- to assign traffic parameters and service category
    -- to the receive and transmit directions of
    -- the ATM virtual links (i.e., VPLs and VCLs).
    -- The ATM VPL or VCL table will indicate a row
    -- in the atmTrafficDescrParamTable
    -- using its atmTrafficDescrParamIndex value.
```

```
     -- The management application can then compare a set of
     -- ATM traffic parameters with a single value.

     -- If no suitable row(s) in the atmTrafficDescrParamTable
     -- exists, the manager must create a new row(s) in this
     -- table. If such a row is created, agent checks the
     -- sanity of that set of ATM traffic parameter values.

     -- The manager may use atmTrafficDescrParamIndexNext
     -- in order to obtain a free atmTrafficDescrParamIndex
     -- value.

     -- When creating a new row, the parameter values
     -- will be checked for self-consistency.
     -- Predefined/template rows may be supported.

     -- A row in the atmTrafficDescrParamTable is deleted
     -- by setting the atmTrafficDescrRowStatus to destroy(6).
     -- The agent will check whether this row is still in use
     -- by any entry of the atmVplTable or atmVclTable.
     -- The agent denies the request if the row is still in
     -- use.

     --  The ATM Traffic Descriptor Parameter Table


     atmTrafficDescrParamTable    OBJECT-TYPE
          SYNTAX       SEQUENCE OF AtmTrafficDescrParamEntry
          MAX-ACCESS   not-accessible
          STATUS       current
          DESCRIPTION
           "This table contains information on ATM traffic
            descriptor type and the associated parameters."
          ::= { atmMIBObjects 5}

     atmTrafficDescrParamEntry    OBJECT-TYPE
          SYNTAX         AtmTrafficDescrParamEntry
          MAX-ACCESS     not-accessible
          STATUS         current
          DESCRIPTION
           "This list contains ATM traffic descriptor
            type and the associated parameters."
          INDEX  {atmTrafficDescrParamIndex}
          ::= { atmTrafficDescrParamTable 1}

     AtmTrafficDescrParamEntry   ::= SEQUENCE  {
       atmTrafficDescrParamIndex  AtmTrafficDescrParamIndex,
       atmTrafficDescrType        OBJECT IDENTIFIER,
```

```
      atmTrafficDescrParam1      Integer32,
      atmTrafficDescrParam2      Integer32,
      atmTrafficDescrParam3      Integer32,
      atmTrafficDescrParam4      Integer32,
      atmTrafficDescrParam5      Integer32,
      atmTrafficQoSClass         INTEGER,
      atmTrafficDescrRowStatus   RowStatus,
      atmServiceCategory         AtmServiceCategory,
      atmTrafficFrameDiscard     TruthValue
                  }

    atmTrafficDescrParamIndex OBJECT-TYPE
         SYNTAX  AtmTrafficDescrParamIndex (1..2147483647)
         MAX-ACCESS     not-accessible
         STATUS         current
         DESCRIPTION
          "This object is used by the virtual link
           table (i.e., VPL or VCL table)
           to identify the row of this table.
           When creating a new row in the table
           the value of this index may be obtained
           by retrieving the value of
           atmTrafficDescrParamIndexNext."
         ::= { atmTrafficDescrParamEntry  1}

    atmTrafficDescrType OBJECT-TYPE
         SYNTAX         OBJECT IDENTIFIER
         MAX-ACCESS     read-create
         STATUS         current
         DESCRIPTION
          "The value of this object identifies the type
           of ATM traffic descriptor.
           The type may indicate no traffic descriptor or
           traffic descriptor with one or more parameters.
           These parameters are specified as a parameter
           vector, in the corresponding instances of the
           objects:
               atmTrafficDescrParam1
               atmTrafficDescrParam2
               atmTrafficDescrParam3
               atmTrafficDescrParam4
               atmTrafficDescrParam5."
          DEFVAL  { atmNoClpNoScr }
          ::= { atmTrafficDescrParamEntry 2}

    atmTrafficDescrParam1 OBJECT-TYPE
         SYNTAX         Integer32
         MAX-ACCESS     read-create
```

```
          STATUS         current
          DESCRIPTION
           "The first parameter of the ATM traffic descriptor
            used according to the value of
            atmTrafficDescrType."
          DEFVAL  { 0 }
          ::= { atmTrafficDescrParamEntry 3}

     atmTrafficDescrParam2 OBJECT-TYPE
          SYNTAX         Integer32
          MAX-ACCESS     read-create
          STATUS         current
          DESCRIPTION
           "The second parameter of the ATM traffic descriptor
            used according to the value of
            atmTrafficDescrType."
          DEFVAL  { 0 }
          ::= { atmTrafficDescrParamEntry 4}

     atmTrafficDescrParam3 OBJECT-TYPE
          SYNTAX         Integer32
          MAX-ACCESS     read-create
          STATUS         current
          DESCRIPTION
           "The third parameter of the ATM traffic descriptor
            used according to the value of
            atmTrafficDescrType."
          DEFVAL  { 0 }
          ::= { atmTrafficDescrParamEntry 5}

     atmTrafficDescrParam4 OBJECT-TYPE
          SYNTAX         Integer32
          MAX-ACCESS     read-create
          STATUS         current
          DESCRIPTION
           "The fourth parameter of the ATM traffic descriptor
            used according to the value of
            atmTrafficDescrType."
          DEFVAL  { 0 }
          ::= { atmTrafficDescrParamEntry 6}

     atmTrafficDescrParam5 OBJECT-TYPE
          SYNTAX         Integer32
          MAX-ACCESS     read-create
          STATUS         current
          DESCRIPTION
           "The fifth parameter of the ATM traffic descriptor
            used according to the value of
```

```
          atmTrafficDescrType."
        DEFVAL  { 0 }
        ::= { atmTrafficDescrParamEntry 7}

    atmTrafficQoSClass  OBJECT-TYPE
        SYNTAX          INTEGER (0..255)
        MAX-ACCESS      read-create
        STATUS          deprecated
        DESCRIPTION
         "The value of this object identifies the QoS Class.
          Four Service classes have been
          specified in the ATM Forum UNI Specification:
          Service Class A: Constant bit rate video and
                           Circuit emulation
          Service Class B: Variable bit rate video/audio
          Service Class C: Connection-oriented data
          Service Class D: Connectionless data
          Four QoS classes numbered 1, 2, 3, and 4 have
          been specified with the aim to support service
          classes A, B, C, and D respectively.
          An unspecified QoS Class numbered `0' is used
          for best effort traffic."
        DEFVAL  { 0 }
        ::= { atmTrafficDescrParamEntry 8}

    atmTrafficDescrRowStatus  OBJECT-TYPE
        SYNTAX          RowStatus
        MAX-ACCESS      read-create
        STATUS          current
        DESCRIPTION
         "This object is used to create
          a new row or modify or delete an
          existing row in this table."
        DEFVAL  { active }
        ::= {atmTrafficDescrParamEntry 9}

    atmServiceCategory  OBJECT-TYPE
        SYNTAX          AtmServiceCategory
        MAX-ACCESS      read-create
        STATUS          current
        DESCRIPTION
         "The ATM service category."
        DEFVAL  { ubr }
        ::= { atmTrafficDescrParamEntry 10}


    atmTrafficFrameDiscard    OBJECT-TYPE
        SYNTAX       TruthValue
```

```
        MAX-ACCESS   read-create
        STATUS       current
        DESCRIPTION
         "If set to 'true', this object indicates that the network
          is requested to treat data for this connection, in the
          given 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."
        DEFVAL { true }
        ::= { atmTrafficDescrParamEntry 11 }

    --  ATM Interface Virtual Path Link (VPL) Table

    -- This table contains configuration and state
    -- information of a bi-directional Virtual Path Link
    -- (VPL)

    -- This table can be used to create, delete or modify
    -- a VPL that is terminated in an ATM host or switch.
    -- This table can also be used to create, delete or
    -- modify a VPL which is cross-connected to another
    -- VPL.

    -- In the example below, the traffic flows on the receive
    -- and transmit directions of the VPLs are characterized
    -- by atmVplReceiveTrafficDescrIndex and
    -- atmVplTransmitTrafficDescrIndex respectively.
    -- The cross-connected VPLs are identified by
    -- atmVplCrossConnectIdentifier.



    --            _____________________________________
    --           |                                     |
    --  VPL      | ATM Host, Switch, or Network |   VPL
    -- receive   |                                     |  receive
    -- ========> X                                    X <=======
    -- <======== X                                    X ========>
    -- transmit  |                                     |  transmit
    --           |_____________________________________|


    --  The ATM Interface VPL Table
```

```
     atmVplTable         OBJECT-TYPE
          SYNTAX         SEQUENCE OF AtmVplEntry
          MAX-ACCESS     not-accessible
          STATUS         current
          DESCRIPTION
           "The Virtual Path Link (VPL) table.  A
            bi-directional VPL is modeled as one entry
            in this table. This table can be used for
            PVCs, SVCs and Soft PVCs.
            Entries are not present in this table for
            the VPIs used by entries in the atmVclTable."
          ::= { atmMIBObjects 6}

     atmVplEntry         OBJECT-TYPE
          SYNTAX         AtmVplEntry
          MAX-ACCESS     not-accessible
          STATUS         current
          DESCRIPTION
           "An entry in the VPL table.  This entry is
            used to model a bi-directional VPL.
            To create a VPL at an ATM interface,
            either of the following procedures are used:

            Negotiated VPL establishment

          (1) The management application creates
              a VPL entry in the atmVplTable
              by setting atmVplRowStatus to createAndWait(5).
              This may fail for the following reasons:
              - The selected VPI value is unavailable,
              - The selected VPI value is in use.
              Otherwise, the agent creates a row and
              reserves the VPI value on that port.

          (2) The manager selects an existing row(s) in the
              atmTrafficDescrParamTable,
              thereby, selecting a set of self-consistent
              ATM traffic parameters and the service category
              for receive and transmit directions of the VPL.

         (2a) If no suitable row(s) in the
              atmTrafficDescrParamTable exists,
              the manager must create a new row(s)
              in that table.

         (2b) The manager characterizes the VPL's traffic
              parameters through setting the
              atmVplReceiveTrafficDescrIndex and the
```

```
            atmVplTransmitTrafficDescrIndex values
            in the VPL table, which point to the rows
            containing desired ATM traffic parameter values
            in the atmTrafficDescrParamTable.  The agent
            will check the availability of resources and
            may refuse the request.
            If the transmit and receive service categories
            are inconsistent, the agent should refuse the
            request.

        (3) The manager activates the VPL by setting the
            the atmVplRowStatus to active(1).
            If this set is successful, the agent has
            reserved the resources to satisfy the requested
            traffic parameter values and the service category
            for that VPL.

        (4) If the VPL terminates a VPC in the ATM host
            or switch, the manager turns on the
            atmVplAdminStatus to up(1) to turn the VPL
            traffic flow on.  Otherwise, the
            atmVpCrossConnectTable  must be used
            to cross-connect the VPL to another VPL(s)
            in an ATM switch or network.

          One-Shot VPL Establishment

          A VPL may also be established in one step by a
          set-request with all necessary VPL parameter
          values and atmVplRowStatus set to createAndGo(4).

          In contrast to the negotiated VPL establishment
          which allows for detailed error checking
          (i.e., set errors are explicitly linked to
          particular resource acquisition failures),
          the one-shot VPL establishment
          performs the setup on one operation but
          does not have the advantage of step-wise
          error checking.

          VPL Retirement

          A VPL is released by setting atmVplRowStatus to
          destroy(6), and the agent may release all
          associated resources."
        INDEX  {ifIndex, atmVplVpi }
        ::= { atmVplTable  1}
```

```
AtmVplEntry    ::= SEQUENCE  {
     atmVplVpi          AtmVpIdentifier,
     atmVplAdminStatus  AtmVorXAdminStatus,
     atmVplOperStatus   AtmVorXOperStatus,
     atmVplLastChange   AtmVorXLastChange,
     atmVplReceiveTrafficDescrIndex
                        AtmTrafficDescrParamIndex,
     atmVplTransmitTrafficDescrIndex
                        AtmTrafficDescrParamIndex,
     atmVplCrossConnectIdentifier INTEGER,
     atmVplRowStatus    RowStatus,
     atmVplCastType     AtmConnCastType,
     atmVplConnKind     AtmConnKind
        }


atmVplVpi           OBJECT-TYPE
     SYNTAX         AtmVpIdentifier
     MAX-ACCESS     not-accessible
     STATUS         current
     DESCRIPTION
      "The VPI value of the VPL."
     ::= { atmVplEntry 1}

atmVplAdminStatus   OBJECT-TYPE
     SYNTAX         AtmVorXAdminStatus
     MAX-ACCESS     read-create
     STATUS         current
     DESCRIPTION
      "This object is instanciated only for a VPL
       which terminates a VPC (i.e., one which is
       NOT cross-co
```