PROJECT: ATM Forum Technical Comittee
SA&A Sub Working Group

**********

SOURCE: SAA API Working Group
Editor: John D. Keene
Interphase Corp.
13800 Senlac
Dallas, TX 75234

Phone: (214)919-9376
Fax: (214)919-9200
E-mail: jkeene@iphase.com

**********

TITLE: Native ATM API Service Description Draft Specification

**********

Date: September 26-30, 1994

**********

ABSTRACT: This document contains the current draft of the Service Description Specification for the Native ATM API. This is the first draft of the Specification. Editor notes and comments are contained within [ ]. These delineate sections that require contributions and/or motions to complete the text.

**********

Distribution List: SA&A SWG

**********

Notice: This contribution has been prepared to assist the ATM Forum. This proposal is made by Interphase Corp. as a basis of discussion. This contribution should not be construed as a binding proposal on Interphase Corp. Specifically, Interphase reserves the right to ammend or modify the statements made herein.

**********

# Native
# ATM API
# Semantic Description
# Specification

Draft Revision 2

# 1. Introduction

## 1.1 Purpose of Document

This document is a specification of the semantic interfaces used between:

- end user or independent software vendor applications,
- new and existing protocol stacks

and physical ATM interfaces installed within end systems as specified in [1]. The document specifies access to the Q.2931 signalling layer, the ATM adaptation layer and the ATM layer.

This document does not specify specific syntactic implementations of the semantic interface nor does it specify where the interface is to exist within an end system. An example reference model is included to show where the interface can be implemented to clarify discussions and uses of the interface.

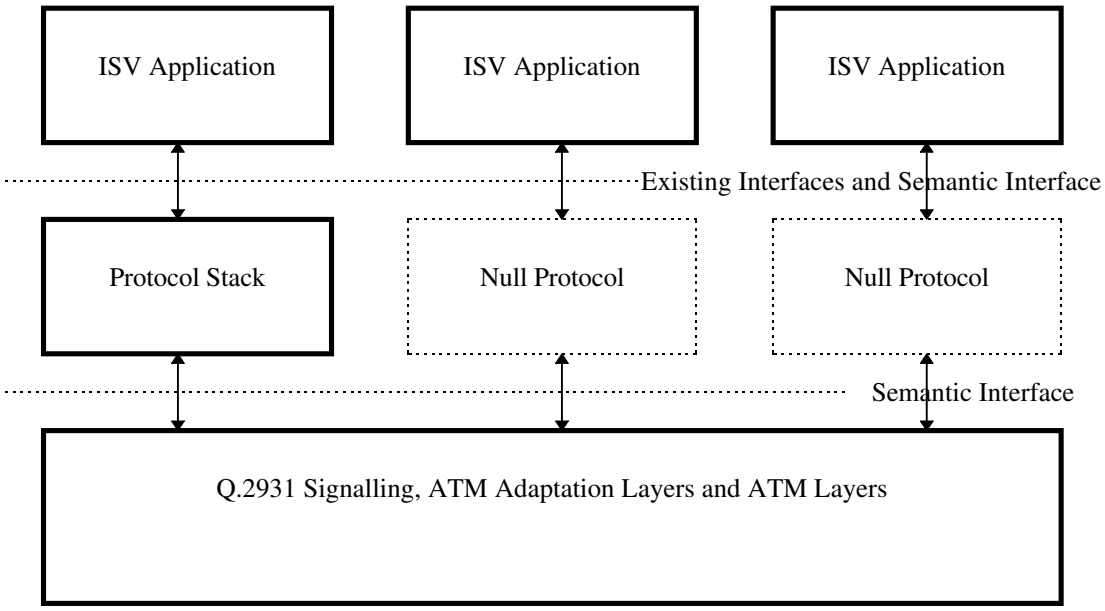
## 1.2 Scope of Document

The scope of the document includes the following:

- A reference model to show uses of the interface.
- The semantic definition of the interface.

## 1.3 Reference Model

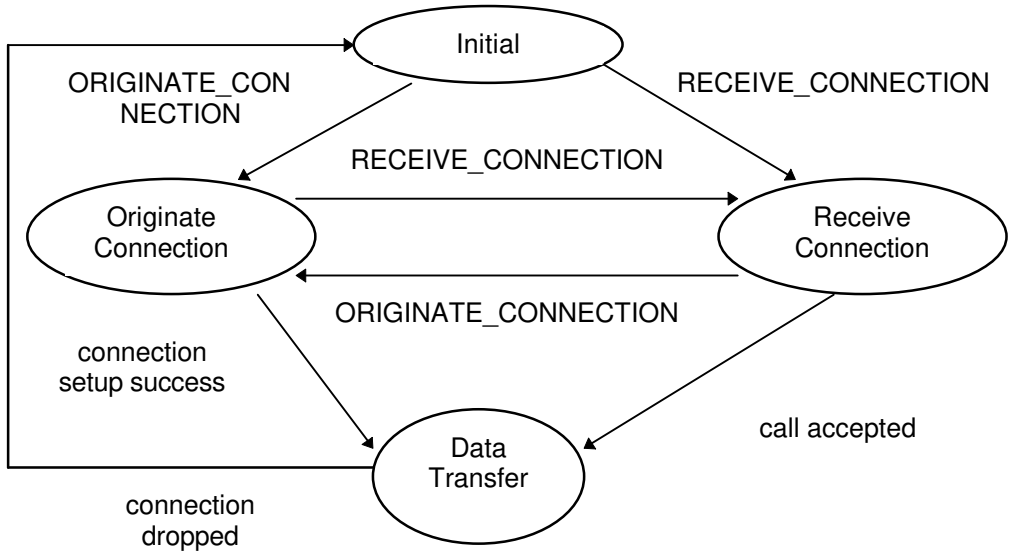
The model shown below depicts a sample implementation of the interface. It does not depict a particular syntactic implementation.




## 1.4 State Diagrams

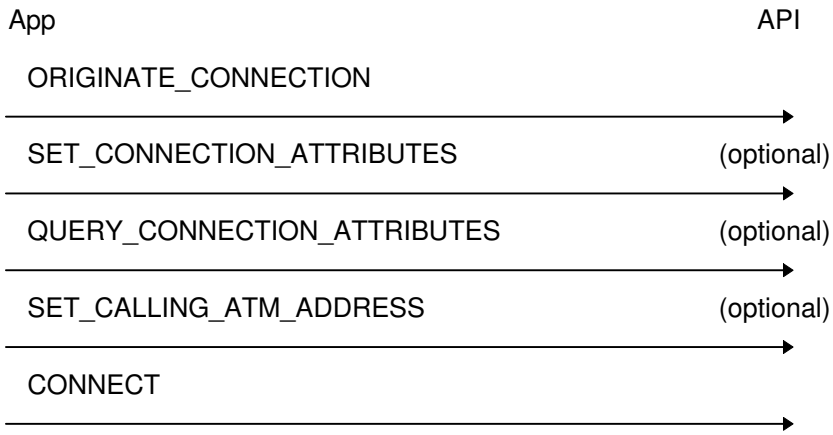
1. Four diagrams are entity-relationship ("E-R") diagrams. The rectangles are entities, the circles are attributes of the attached entities, and the diamonds are relationships between the entities. The numbers on either side of the diamond denotes the cardinality of the relationship (e.g. one-to-one, many-to-one).
2. There are three finite-state-machine ("FSM") diagrams. The circles are the states and the edges are the transitions between states.
3. According to the diagrams, there are 4 top-level states to the API: Initial, Originate_Connection, Receive_Connection, and Data_Transfer.

### 1.4.1 Top Level State Diagram



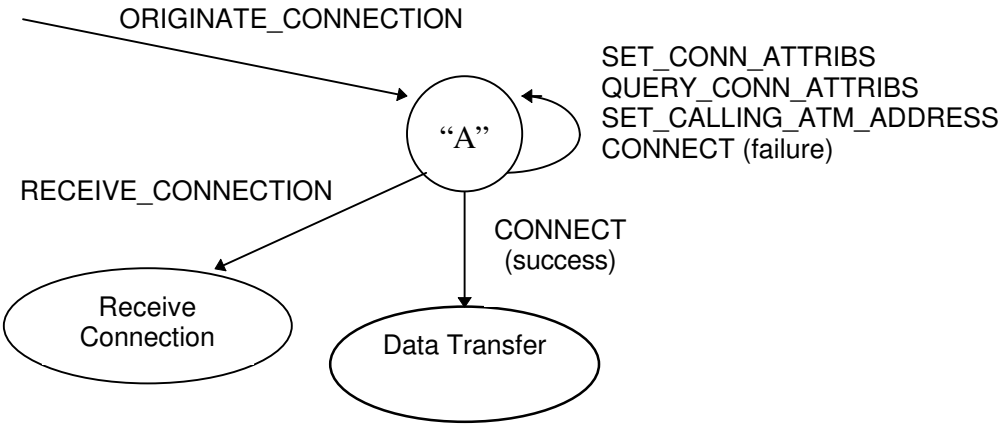

1. An application is free to switch between an intention to receive or originate a call via the primitives RECEIVE_CONNECTION and ORIGINATE_CONNECTION.

### 1.4.2 Originate Connection Flows

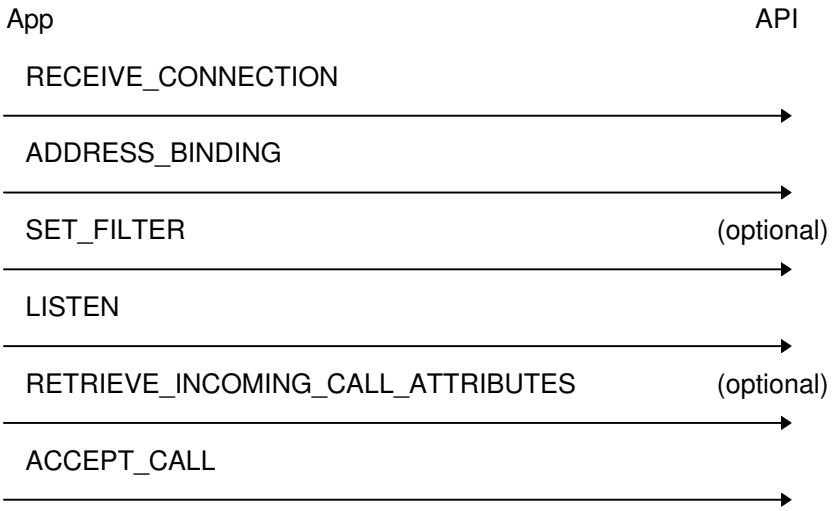



### 1.4.3 Originate Connection FSM

1. The states inside the cloud (i.e. "A") represent a more detailed FSM for the state "Originate_Connection" on the top-level FSM.

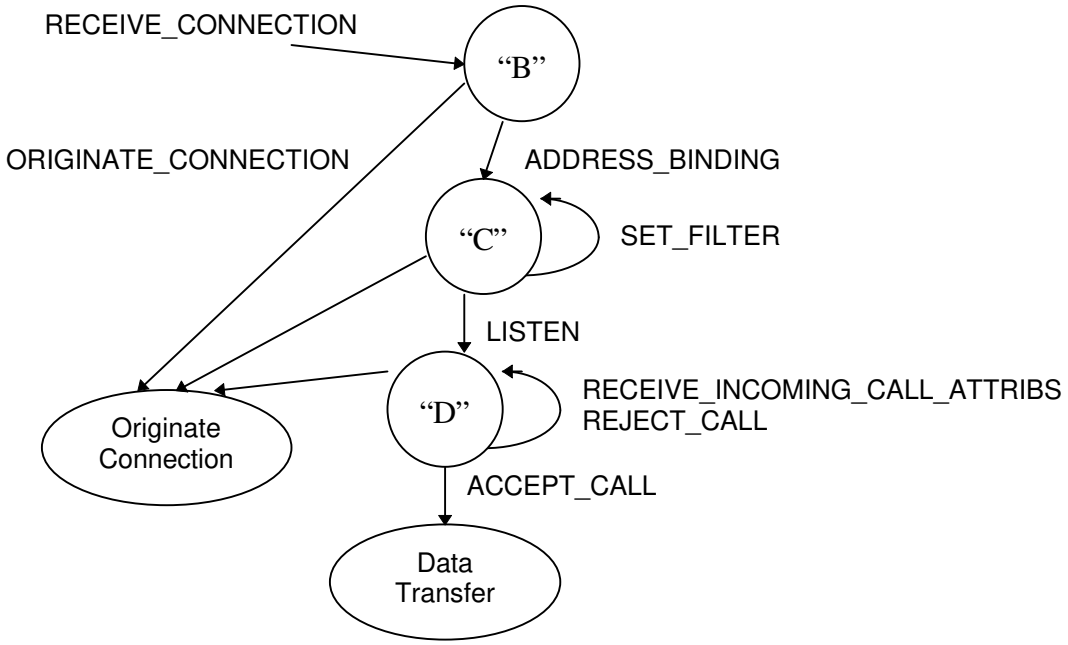



### 1.4.4 Receive Connection Flows



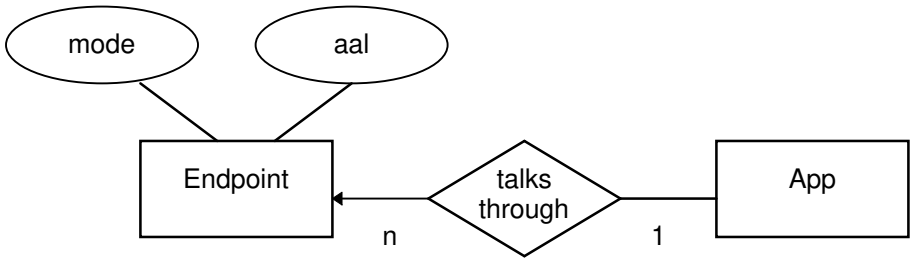

### 1.4.5 Receive Connection FSM

The states inside the cloud (i.e. "B", "C", "D") are a more detailed FSM for the state "Receive_Connection" on the top-level FSM.




## 1.4.6 Initial Mode E-R

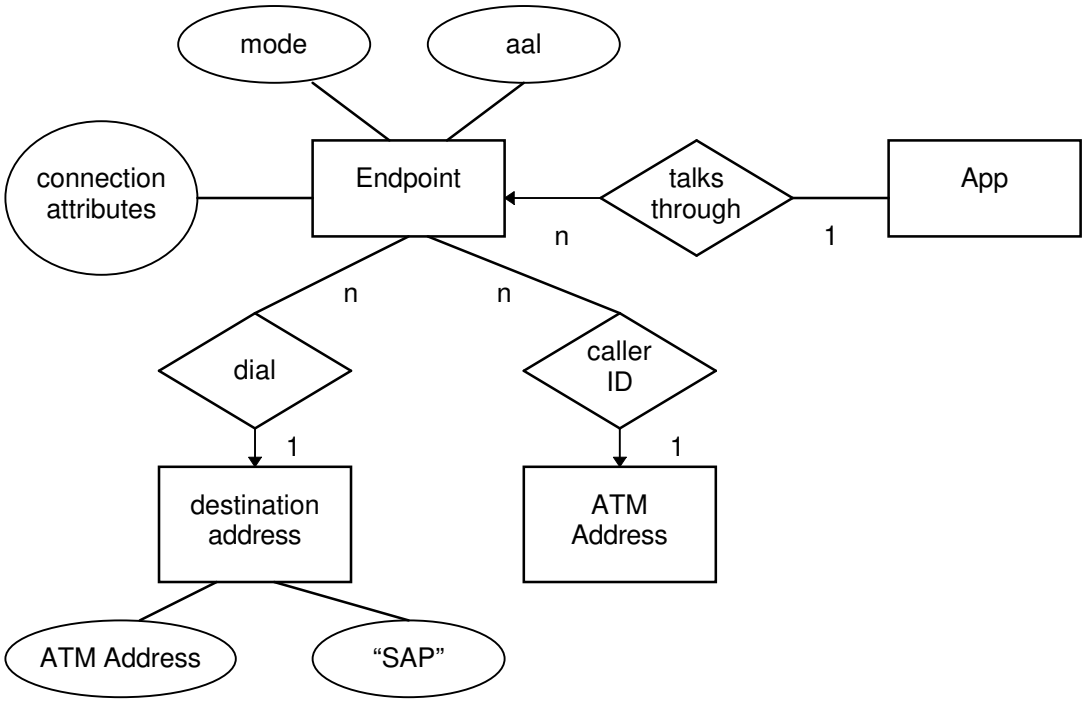
1. This is the state of the API after the application has issued ATM_API_CREATE_ENDPOINT.
2. Each application may talk thru many endpoints, but each endpoint is associated with a unique application.




## 1.4.7 Originate Connection Mode E-R

1. This is the state of the API corresponding to an application originating a call. The primitive is called ORIGINATE_CONNECTION.
2. The application may have optionally issued the primitives ATM_API_SET_CONNECTION_ATTRIBUTE and ATM_API_QUERY_CONNECTION_ATTRIBUTE to manipulate the connection attributes.
3. The application may have optionally specified the ATM address that this call is being originated from (if the originating station supports multiple ATM addresses). The primitive for this is called SET_CALLING_ATM_ADDRESS. Note that this is semantically different from ATM_API_BIND_ENDPOINT. Many endpoints may specifiy the same calling ATM address.

4. The application must specify the destination address during the ATM_API_CONNECT. This destination address is composed of the ATM address of the target station, plus a "SAP" (service access point) inside of the target station. Many endpoints may specify the same destination address.

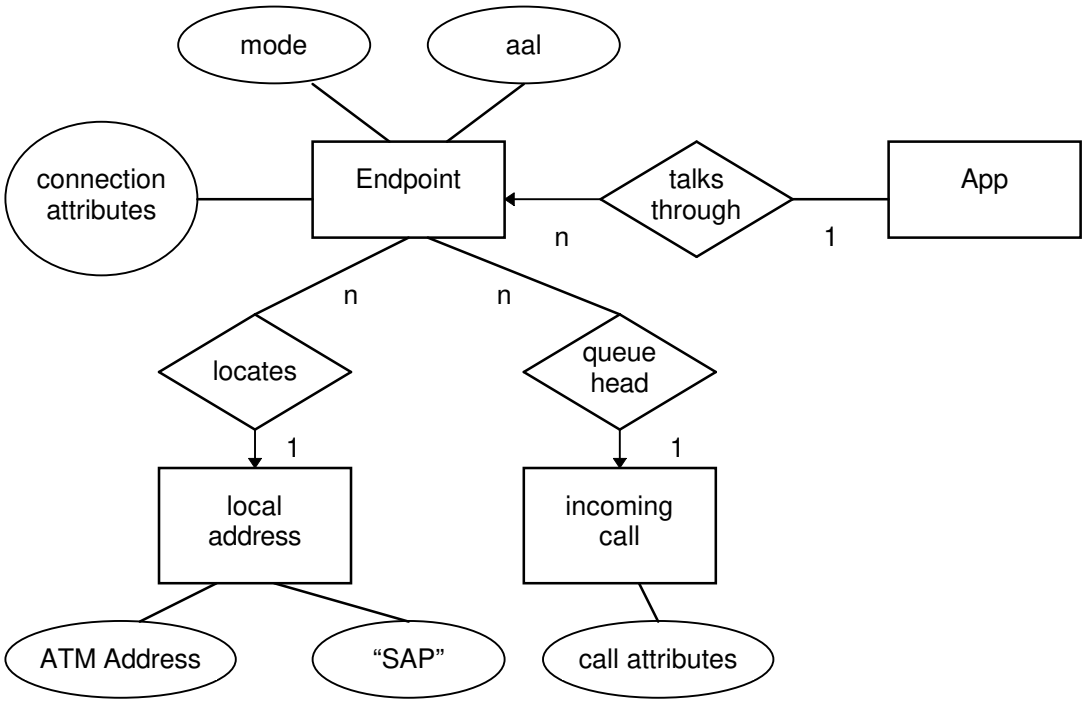



## 1.4.8 Receive Connection Mode E-R

1. This is the state of the API corresponding to an application receiving a call. The primitive is called RECEIVE_CONNECTION.
2. The application sets the receive queue size via the primitive ATM_API_LISTEN.
3. The application may have optionally set filter attributes via the primitive ATM_API_SET_FILTER.
4. The application must specify a unique local address to be associated in a one-to-one relationship with the endpoint.

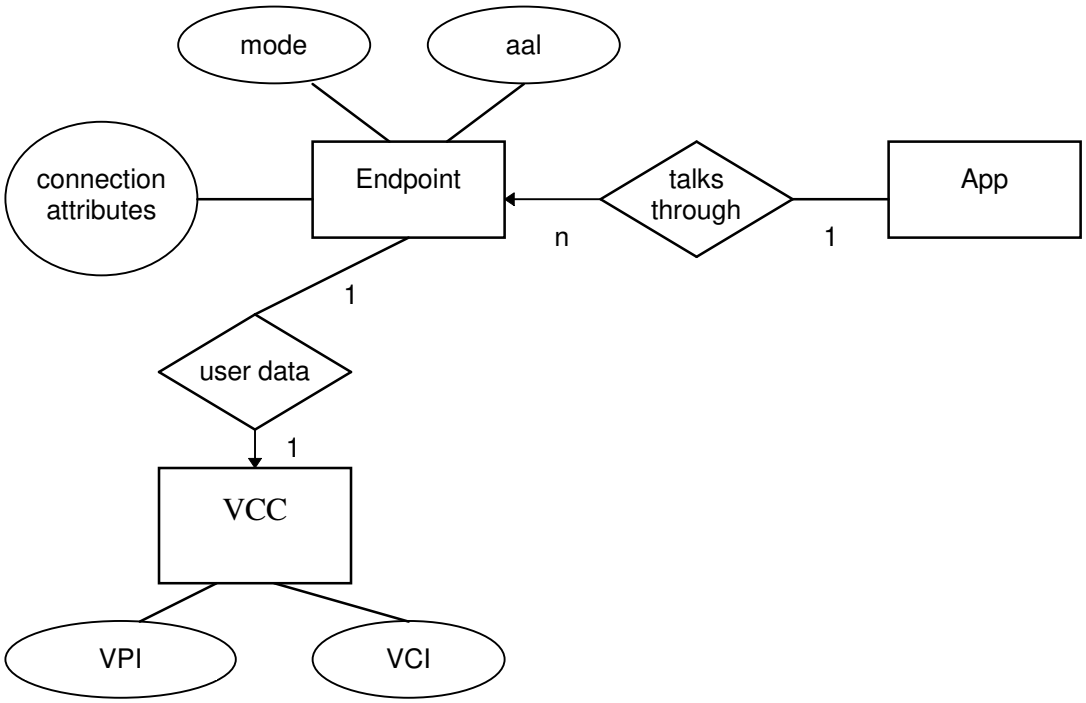
The local address is composed of the ATM address of this station, plus a "SAP" (service access point) inside of this station. This is accomplished via the primitive ATM_API_BIND_ENDPOINT.

5. For each endpoint, there is a unique queue holding incoming call requests. The application can examine attributes of the incoming call request at the head of the queue, via the primitive ATM_API_RETRIEVE_INCOMING_CALL_ATTRIBUTE.




## 1.4.9 Data Transfer Mode E-R

1. This state is reached by one of two ways:
   a). the call that this station was attemping to originate was successfully completed
   b). the incoming call was accepted by this station.
2. The connection attributes in this state were inherited in one of two ways:
   a). if the station was originating the call, then the attibutes being set and/or queried by the application are inherited
   b). if the station was accepting an incoming call, then the attributes of the specific call accepted are inherited.
3. There is a one-to-one correspondence between an endpoint and a VCC (virtual connection circuit). Each VCC may be identified on the ATM wire as having a VPI (virtual path identifier) and VCI (virtual connection identifier).




## 1.5 Reference Documents

The following documents are referred to within this document. They define various aspects of the ATM layers that are being abstracted.

I.361 - B-ISDN ATM layer specification
I.362 - B-ISDN ATM Adaptation Layer (AAL) Functional Description
I.363 - B-ISDN ATM Adaptation Layer (AAL) Specification
Q.2100 - ???

# 2. Semantic Description

## 2.1 API Initialization

Immediately after an ATM application is running, it needs to register itself to the ATM API. Before the application is terminated, it needs to de-register itself from th ATM API.

### 2.1.1 Initialize API

This service establishes the communication between an ATM application and the ATM API. This should be the first call of each application that wants to access the ATM API.

The semantics of this service are as follows:

**ATM_API_INITIALIZE (**

**)**

During initialization or registration, the following functions are performed if necessary:
- verify if AAL/ATM service is loaded and operable,
- ATM API version negotiation,
- initialize any per application control data structure

If any of the above functions fail, this service returns an error indicating the cause and the application is unable to utilize the ATM API services.

There are no required parameters for this service.

### 2.1.2 Terminate API

This service is used by the application to de-register from the ATM API entity.

The semantics of this service are as follows:

**ATM_API_TERMINATE (**

**)**

There are no required parameters for this service. This service always returns success.

## 2.2 Communication Endpoint

A communication endpoint is defined as an object associated with a set of attributes which can be specified at the communication creation time. This communication endpoint will be used in the subsequent ATM API services.

### 2.2.1 Create a Communication Endpoint

This service is used to create an ATM communication endpoint with specified attributes.

The semantics of this service are as follows:

**ATM_API_CREATE_ENDPOINT (**
**IN** *type_of_aal* **,**
**IN** *communication_mode* **,**
**OUT** *communication_endpoint*
**)**

The type_of_aal parameter specifies which AAL type will be used on this communication endpoint, e.g., AAL0, AAL1, AAL2, AAL3/4 or AAL5. The communication_mode specifies which service mode, e.g., stream mode or message mode, assured or non-assured mode, will be used on this communication endpoint. The communication_endpoint parameter is the unique handle to the created communication endpoint.

### 2.2.2 Address Binding

This service allows applications to associate a communication endpoint with a local address.

The semantics of this service are as follows:

**ATM_API_BIND_ENDPOINT (**
**IN** *communication_endpoint* **,**
**IN** *local_address*
**)**

Note that by local address, we mean an address that can be used to identify a unique communication endpoint, not necessarily a physical ATM address.

### 2.2.3 Close a Communication Endpoint

This service is used to terminate a communication endpoint. All connections associated with this communication endpoint will be aborted and all the resources allocated for the communication endpoint are released.

The semantics of this service are as follows:

**ATM_API_TERMINATE_ENDPOINT (**
    **IN** *communication_endpoint*
**)**

## 2.3 Call/Connection Control

### 2.3.1 Point-to-Point Connection

#### 2.3.1.1 Call/Connection Setup

##### 2.3.1.1.1 Set/Query Call/Connection Attributes

These services allow applications to set/get connection attributes.

The semantics of this service are as follows:

**ATM_API_SET_CONNECTION_ATTRIBUTES (**
    **IN** *communication_endpoint* **,**
    **IN** *attribute_type* **,**
    **IN** *data*
**)**

The attribute_type parameter specifies the type of attribute, e.g., QoS, bandwidth requirements, user data. The data parameter specifies the value of the attribute.

**ATM_API_QUERY_CONNECTION_ATTRIBUTES (**
    **IN** *communication_endpoint* **,**
    **IN** *attribute_type* **,**
    **OUT** *data*
**)**

The attribute_type parameter specifies the type of attribute being queried, e.g., QoS, bandwidth requirements, or user data. The data parameter is the query result.

##### 2.3.1.1.2 Connect

This service allows applications to make a connection request to a destination address.

The semantics of this service are as follows:

**ATM_API_CONNECT (**
    **IN** *communication_endpoint* **,**

**IN** *destination_address*
**)**

The destination_address parameter specifies a remote communication endpoint.

### 2.3.1.2 Incoming Call/Connection Process

#### 2.3.1.2.1 Set Call/Connection Filter

This service allows applications, at the operating system level, to set a filter for screening incoming calls/connections with certain connection attributes.

The semantics of this service are as follows:

**ATM_API_SET_FILTER (**
**IN** *communication_endpoint* **,**
**IN** *attribute_type* **,**
**IN** *data*
)

The attribute_type parameter specifies the type of attribute. The data parameter specifies the value of the attribute.

#### 2.3.1.2.2 Listen

This service allows applications to set the incoming queue size of a communication endpoint. A communication endpoint will be ready to accept incoming calls after this service call.

The semantics of this service are as follows:

**ATM_API_LISTEN (**
**IN** *communication_endpoint* **,**
**IN** *queue_size*
)

The queue_size parameter specifies the maximum number of incoming calls allowed to wait in the queue.

#### 2.3.1.2.3 Retrieve Incoming Call Attributes

This service allows applications to retrieve incoming call's attributes, e.g., QoS requests, caller's ID, with which the application can decide to accept or reject this call.

The semantics of this service are as follows:

**ATM_API_RETRIEVE_INCOMING_CALL_ATTRIBUTE (**
**IN** *communication_endpoint* **,**
**OUT** *incoming_call_attribute*
**)**

### 2.3.1.2.4 Reject an Incoming Call/Connection

The service allows applications to reject an incoming call request. Applications may specify the cause for the rejection.

The semantics of this service are as follows:

**ATM_API_REJECT_CALL (**
**IN** *communication_endpoint* **,**
**IN** *reject_cause*
**)**

### 2.3.1.2.5 Accept an Incoming Call/Connection

This service allows applications to accept an incoming call request. A new communication endpoint is created and linked to a VCC connected to the remote end.

The semantics of this service are as follows:

**ATM_API_ACCEPT_CALL (**
**IN** *communication_endpoint* **,**
**OUT** *new_communication_endpoint*
**)**

### 2.3.1.3 Call/Connection Disconnect

This service allows applications to disconnect a connection associated with a communication endpoint.

The semantics of this service are as follows:

**ATM_API_DISCONNECT_ENDPOINT (**
**IN** *communication_endpoint* **,**
**IN** *attribute_type* **,**
**IN** *data*
**)**

The attribute_type parameter specifies the type of attribute, e.g., disconnect method and user data. The parameter data specifies the value of the attribute, e.g., graceful disconnect or aborted disconnect for disconnect method.

### 2.3.2 Point-to-Multipoint

#### 2.3.2.1 Add a Party

This service allows applications to add a new leaf to a point-to-multipoint connection. This can only be initiated at the root node which originates the first point-to-point connection. Note that based on [1], only non-zero identical bandwidth from rote node to each leaf node and zero bandwidth from each leaf node to the root node are supported. Therefore, no QoS is specified in this service. Also note that traffic in a point-to-multipoint connection only flows from the root node to each leaf node.

The semantics of this service are as follows:

**ATM_API_ADD_PARTY (**
    **IN** *communication_endpoint* **,**
    **IN** *destination_address*
)

#### 2.3.2.2 Drop a Party

This service allows applications to drop a leaf node from a point-to-multipoint connection. This can be initiated either from a leaf or a root node.

The semantics of this service are as follows:

**ATM_API_DROP_PARTY (**
    **IN** *communication_endpoint* ,
    **IN** *destination_endpoint*
)

### 2.3.3 Multicast

#### 2.3.3.1 Join a Group

This service is used for a communication endpoint to join a multicast group.

The semantics of this service are TBD.

#### 2.3.3.2 Leave a Group

This service is used for a communication endpoint to leave a multicast group.

The semantics of this service are TBD.

## 2.4 Data Transmission

### 2.4.1 Send

This service is generated by the application layer to send data to a remote communication endpoint or a group of communication endpoints in the case of point-to-multipoint connections.

The semantics of this service are as follows:

**ATM_API_SEND_DATA (**
    **IN** *communication_endpoint* **,**
    **IN** *data* **,**
    **OUT** *byte_count*
**)**

The byte_count parameter specifies the number of bytes sent. In the case of stream mode, the value of byte_count might be less than or equal to the length of data to be sent, e.e., only the first byte_count bytes of data were transmitted. This applies to AAL types which support stream mode.

In the case of message mode, if the value of byte_count differs from the length of data, this indicates an error which occurs during data transmission. This applies to AAL types which support message mode.

### 2.4.2 Receive

This service indicates the arrival of user data. The semantics of this service are as follows:

**ATM_API_RECEIVE_DATA (**
    **IN** *communication_endpoint* ,
    **OUT** *data* ,
    **OUT** *byte_count*
**)**

In the case of stream mode, the byte_count parameter is the number of bytes received. In the case of message mode, the message boundaries are preserved.

## 2.5 Management Interface

[Editor's Note: This will need to explicitly delineate what is required to be available what may be priveleged and thus have restricted access.]

### 2.5.1 Service Initialization and Termination

### 2.5.2 Service Configuration

### 2.5.3 F4/F5 Flows

### 2.5.4 Error Indications

### 2.5.5 Query Capabilities

## 2.6 SSCOP Service Access Points

[Editor's Note: We may want to include SSCOP Service Access Points within this document. Actual inclusion will be determined by contributions. Q.2110 or Q.2100]

# References

[1] “ATM User-Network Interface Specification Version 3.0”, ATM Forum, September 1993