The ATM Forum
Technical Committee

Native ATM Services
Data Link Provider Interface
(DLPI) Addendum
Version 1.0

AF-SAA-API-DLPI-0091.000

February, 1998

© 1998 by The ATM Forum. The ATM Forum hereby grants its members the limited right to reproduce in whole, but not in part, this specification for its members internal use only and not for further distribution. This right shall not be, and is not, transferable. All other rights reserved. Except as expressly stated in this notice, no part of this document may be reproduced or transmitted in any form or by any means, or stored in any information storage and retrieval system, without the prior written permission of The ATM Forum.

The information in this publication is believed to be accurate as of its publication date. Such information is subject to change without notice and The ATM Forum is not responsible for any errors. The ATM Forum does not assume any responsibility to update or correct any information in this publication. Notwithstanding anything to the contrary, neither The ATM Forum nor the publisher make any representation or warranty, expressed or implied, concerning the completeness, accuracy, or applicability of any information contained in this publication. No liability of any kind shall be assumed by The ATM Forum or the publisher as a result of reliance upon any information contained in this publication.

The receipt or any use of this document or its contents does not in any way create by implication or otherwise:

- Any express or implied license or right to or under any ATM Forum member company's patent, copyright, trademark or trade secret rights which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor

- Any warranty or representation that any ATM Forum member companies will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor

- Any form of relationship between any ATM Forum member companies and the recipient or user of this document.

Implementation or use of specific ATM standards or recommendations and ATM Forum specifications will be voluntary, and no company shall agree or be obliged to implement them by virtue of participation in The ATM Forum.

The ATM Forum is a non-profit international organization accelerating industry cooperation on ATM technology. The ATM Forum does not, expressly or otherwise, endorse or promote any specific products or services.

NOTE: The user's attention is called to the possibility that implementation of the ATM interoperability specification contained herein may require use of an invention covered by patent rights held by ATM Forum Member companies or others. By publication of this ATM interoperability specification, no position is taken by The ATM Forum with respect to validity of any patent claims or of any patent rights related thereto or the ability to obtain the license to use such rights. ATM Forum Member companies agree to grant licenses under the relevant patents they own on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. For additional information contact:

The ATM Forum
Worldwide Headquarters
2570 West El Camino Real, Suite 304
Mountain View, CA 94040-1313
Tel: +1-650-949-6700
Fax: +1-650-949-6705

The ATM Forum wishes to especially thank Serge Sasyan who served as the editor and contributed much to this document.

James J. Harford
Chairman, SAA/API Work Group

1. INTRODUCTION 1

1.1 Referenced documents 1

1.2 Glossary 2
1.2.1 The ATM Forum Glossary (subset of terms used in this document) 2
1.2.2 Data Link Provider Interface (DLPI) Glossary 8

2. PROPOSED ARCHITECTURE 10

3. ATM PROVIDER DLPI ADDENDUM 11

3.1 Native ATM Services Data Link Provider Specific Addendum 11
3.1.1 DLSAP Address space 11
3.1.1.1 DLSAP format 11
3.1.1.1.1 SVE tagging 12
3.1.1.1.2 SVE encoding 12
3.1.1.1.2.1 ATM_addr 13
3.1.1.1.2.2 ATM_selector 14
3.1.1.1.2.3 Tagging 14
3.1.1.1.2.4 BLLI_id2 14
3.1.1.1.2.5 BLLI_id3 16
3.1.1.1.2.6 BHLI_id 17
3.1.1.2 DLSAP registration 17
3.1.2 PPA Access and Control 22
3.1.3 Quality of Service 22
3.1.4 DL_INFO_ACK Values 23
3.1.5 Supported Services 23

4. MAPPING OF PRIMITIVES 25

4.1 Mapping of Native ATM Services primitives to ATM DLPI 25
4.1.1 State mapping 25
4.1.2 Primitives mapping table 26
4.1.3 Primitives parameters general mapping 28

5. ATM HEADER FILE 34

1. Introduction

DLPI (Data Link Provider Interface) Specification is a Common Applications Environment (CAE) Specification (Copyright February 1997, the Open Group) of X/Open. Appendix E of this DLPI Specification defines the information that are required to produce a Data Link Service provider-specific Addendum, i.e.:

- the DLSAP Address space,
- the PPA Access and Control,
- the Quality of Service,
- the DL_INFO_ACK Values,
- and the Supported Services.

This document is the Addendum required in "CAE specification Data Link Provider Interface (DLPI)" Appendix E for a given Data Link Service Provider. It defines the part of the Native ATM Services, described in the ATM Forum document af-saa-0048.000 "Native ATM Services : Semantic Description Version 1.0", that can be supported by DLPI.

This document "Native ATM Services Data Link Provider Specific Addendum" explains the current DLPI options for accessing ATM connection attributes ; an included header file defines the mapping between DLPI data structures and the Q.2931 information elements.

DLPI allows kernel space link level software components to access ATM services. It is an API suitable for Classical IP or LAN Emulation, or other kernel space software components over ATM. DLPI provides access to user space applications as well.

According to the guidelines in Appendix E of the DLPI specifications, this document is proposing a definition of the ATM specific Addendum to the Data Link Provider Interface Specification. Herein, any data link provider conforming to this specification is called an "ATM DLS Provider". Any application using the ATM Data Link Services through the Data Link Provider Interface according to this specification is called an "ATM DLS User".

1.1 Referenced documents

- af-saa-0048.000 "Native ATM Services : Semantic Description Version 1.0", ATM Forum.
- CAE (Common Applications Environment) Specification, Data Link Provider Interface (DLPI), (Copyright February 1997, the Open Group), Document number: C614, ISBN: 1-85912-196-9.
- "User-Network Interface specification version 3.1", ATM Forum.
- ITU-T Q.2931 B-ISDN DSS2 User-Network Interface (UNI) Layer 3

Specification for Basic Call/Connection Control.

- ITU-T Q.2931 (02/95) Digital Subscriber Signalling System No. 2 (DSS 2) – User-Network Interface (UNI) layer 3 specification for basic call/connection control, Modified by Rec. Q.2971 (10/1995) 248pp E 6312 F 6313 S 6314 80 CHF
- ITU-T Q.2931 Amendment 1 (06/97) Digital Subscriber Signalling System No. 2 (DSS 2) – User-Network Interface (UNI) layer 3 specification for basic call/connection control, To be published...

1.2 Glossary

The acronyms used in this document are related either to ATM or to DLPI. Here is the subset of the ATM Forum list of acronyms used in this document and the acronyms related to DLPI as defined in the Glossary inside the DLPI specification.

1.2.1 The ATM Forum Glossary (subset of terms used in this document)

AAL
: ATM Adaptation Layer: The standards layer that allows multiple applications to have data converted to and from the ATM cell. A protocol used that translates higher layer services into the size and format of an ATM cell.

AAL Connection
: Association established by the AAL between two or more next higher layer entities.

AAL-1
: ATM Adaptation Layer Type 1: AAL functions in support of constant bit rate, time-dependent traffic such as voice and video.

AAL-5
: ATM Adaptation Layer Type 5: AAL functions in support of variable bit rate, delay-tolerant connection-oriented data traffic requiring minimal sequencing or error detection support.

API
: Application Program Interface: API is a programmatic interface used for interprogram communications or for interfacing between protocol layers.

API_connection
: Native ATM Application Program Interface Connection: API_connection is a relationship between an API_endpoint and other ATM devices that has the following characteristics:
  - Data communication may occur between the API_endpoint and the other ATM devices comprising the API_connection.
  - Each API_connection may occur over a duration of time only once; the same set of communicating ATM devices may form a new connection after a prior connection is released.
  - The API_connection may be presently active (able to transfer data), or merely anticipated for the future.

ATM
: Asynchronous Transfer Mode: A transfer mode in which the information is organized into cells. It is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic.

ATM Address
: Defined in the UNI Specification as 3 formats, each having 20 bytes in length including country, area and end-system identifiers.

ATM Layer Link
: A section of an ATM Layer connection between two adjacent active ATM Layer entities (ATM-entities).

ATM Traffic Descriptor
: A generic list of traffic parameters that can be used to capture the intrinsic traffic characteristics of a requested ATM connection.

ATM User-User Connection
: An association established by the ATM Layer to support communication between two or more ATM service users (i.e., between two or more next higher entities or between two or more ATM-entities). The communications over an ATM Layer connection may be either bidirectional or unidirectional. The same Virtual Channel Identifier (VCI) issued for both directions of a connection at an interface.

B-ISDN
: Broadband ISDN: A high-speed network standard (above 1.544 Mbps) that evolved Narrowband ISDN with existing and new services with voice, data and video in the same network.

B-LLI
: Broadband Low Layer Information: This is a Q.2931 information element that identifies a layer 2 and a layer 3 protocol used by the application.

BCOB
: Broadband Connection Oriented Bearer: Information in the SETUP message that indicates the type of service requested by the calling user.

BCOB-A
: Bearer Class A: Indicated by ATM end user in SETUP message for connection-oriented, constant bit rate service. The network may perform internetworking based on AAL information element (IE).

BCOB-C
: Bearer Class C: Indicated by ATM end user in SETUP message for connection-oriented, variable bit rate service. The network may perform internetworking based on AAL information element (IE).

BCOB-X
: Bearer Class X: Indicated by ATM end user in SETUP message for ATM transport service where AAL, traffic type and timing requirements are transparent to the network.

BHLI
: Broadband High Layer Information: This is a Q.2931 information element that identifies an application (or session layer protocol of an application).

Broadband
: A service or system requiring transmission channels capable of supporting rates greater than the Integrated Services Digital Network (ISDN) primary rate.

Call
: A call is an association between two or more users or between a user and a network entity that is established by the use of network capabilities. This association may have zero or more connections.

CBR
: Constant Bit Rate: An ATM service category which supports a constant or guaranteed rate to transport services such as video or voice as well as circuit emulation which requires rigorous timing control and performance parameters.

Connection
: An ATM connection consists of concatenation of ATM Layer links in order to provide an end-to-end information transfer capability to access points.

Connectionless
: Refers to ability of existing LANs to send data without previously establishing connections.

DA
: Destination Address: Information sent in the forward direction indicating the address of the called station or customer.

DLPI
: UNIX International, Data Link Provider Interface (DLPI) Specification: Revision 2.0.0, OSI Work Group, August 1991.

DTE
: Data Terminal Equipment: A generic definition of external networking interface equipment such as a modem.

E.164
: A public network addressing standard utilizing up to a maximum of 15 digits. ATM uses E.164 addressing for public network addressing.

ESI
: End System Identifier: This identifier distinguishes multiple nodes at the same level in case the lower level peer group is partitioned.

FEC
: Forward Error Correction: A technique for detection and correction of errors in a digital data stream.

GCRA
: Generic Cell Rate Algorithm: The GCRA is used to define conformance with respect to the traffic contract of the connection. For each cell arrival the GCRA determines whether the cell conforms to the traffic contract. The UPC function may implement the GCRA, or one or more equivalent algorithms to enforce conformance. The GCRA is defined with two parameters: the Increment (I) and the Limit (L).

HDLC
: High Level Data Link Control: An ITU-TSS link layer protocol standard for point-to-point and multi-point communications.

Header
: Protocol control information located at the beginning of a protocol data unit.

IEC
: Inter-exchange Carrier: A long distance telephone company.

IEEE
: Institute of Electrical and Electronics Engineers: A worldwide engineering publishing and standards-making body for the electronics industry.

IEEE 802.3
: A Local Area Network protocol suite commonly known as Ethernet. Ethernet has either a 10 Mbps or 100 Mbps throughput and uses Carrier Sense Multiple Access bus with Collision Detection CSMA/CD. This method allows users to share the network cable. However, only one station can use the cable at a time. A variety of physical medium dependent protocols are supported.

IEEE 802.5
: A Local Area Network protocol suite commonly known as Token Ring. A standard originated by IBM for a token passing ring network that can be configured in a star topology. Versions supported are 4 Mbps and

16 Mbps.

ILMI

Integrated Local Managment Interface: An ATM Forum defined interim specification for network management functions between an end user and a public or private network and between a public network and a private network. This is based on a limited subset of SNMP capabilities.

IP

Internet Protocol: Originally developed by the Department of Defense to support interworking of dissimilar computers across a network. This protocol works in conjunction with TCP and is usually identified as TCP/IP. A connectionless protocol that operates at the network layer (layer 3) of the OSI model.

IS

Intermediate System: A system that provides forwarding functions or relaying functions or both for a specific ATM connection. OAM cells may be generated and received.

ISO

International Organization for Standardization: An international organization for standardization, based in Geneva, Switzerland, that establishes voluntary standards and promotes global trade of 90 member countries.

ITU-T

International Telecommunications Union Telecommunications: ITU-T is an international body of member countries whose task is to define recommendations and standards relating to the international telecommunications industry. The fundamental standards for ATM have been defined and published by the ITU-T (Previously CCITT).

LAN

Local Area Network: A network designed to move data between stations within a campus.

Link

An entity that defines a topological relationship (including available transport capacity) between two nodes in different subnetworks. Multiple links may exist between a pair of subnetworks. Synonymous with logical link.

Link Attribute

A link state parameter that is considered individually to determine whether a given link is acceptable and/or desirable for carrying a given connection.

MAC

Media Access Control: IEEE specifications for the lower half of the data link layer (layer 2) that defines topology dependent access control protocols for IEEE LAN specifications.

MBS

Maximum Burst Size: In the signaling message, the Burst Tolerance (BT) is conveyed through the MBS which is coded as a number of cells. The BT together with the SCR and the GCRA determine the MBS that may be transmitted at the peak rate and still be in conformance with the GCRA.

MIB

Management Information Base: A definition of management items for some network component that can be accessed by a network manager. A MIB includes the names of objects it contains and the type of information retained.

NSAP

Network Service Access Point: OSI generic standard for a network address consisting of 20 octets. ATM has specified E.164 for public network addressing and the NSAP address structure for private network addresses.

Octet

A term for eight (8) bits that is sometimes used interchangeably with "byte" to mean the same thing.

OUI

Organizationally Unique Identifier: The OUI is a three-octet field in the IEEE 802.1a defined SubNetwork Attachment Point (SNAP) header, identifying an organization which administers the meaning of the following two octet Protocol Identifier (PID) field in the SNAP header. Together they identify a distinct routed or bridged protocol.

PCR

Peak Cell Rate: The Peak Cell Rate, in cells/sec, is the cell rate which the source may never exceed.

PDU

Protocol Data Unit: A PDU is a message of a given protocol comprising payload and protocol-specific control information, typically contained in a header. PDUs pass over the protocol interfaces which exist between the layers of protocols (per OSI model).

PID

Protocol Identification. Refer to OUI.

Primitive

An abstract, implementation independent, interaction between a layer service user and a layer service provider.

Protocol

A set of rules and formats (semantic and syntactic) that determines the communication behavior of layer entities in the performance of the layer functions.

Protocol Control Information

Information exchanged between corresponding entities, using a lower layer connection, to coordinate their joint operation.

PVC

Permanent Virtual Circuit: This is a link with static route defined in advance, usually by manual setup.

QoS

Quality of Service: Quality of Service is defined on an end-to-end basis in terms of the following attributes of the end-to-end ATM connection:

- Cell Loss Ratio,
- Cell Transfer Delay,
- Cell Delay Variation.

SAP

Service Access Point: A SAP is used for the following purposes:

- When the application initiates an outgoing call to a remote ATM device, a destination_SAP specifies the ATM address of the remote device, plus further addressing that identifies the target software entity within the remote device,
- When the application prepares to respond to incoming calls from remote ATM devices, a local_SAP specifies the ATM address of the device housing the application, plus further addressing that identifies the application within the local device.

There are several groups of SAPs that are specified as valid for Native ATM Services.

SCR

Sustainable Cell Rate: The SCR is an upper bound on the conforming average rate of an ATM connection over time scales which are long relative to those for which the PCR is defined. Enforcement of this bound by the UPC could allow the network to allocate sufficient resources, but less than those based on the PCR, and still ensure that the performance objectives (e.g., for Cell Loss Ratio) can be achieved.

SDU

Service Data Unit: A unit of interface information whose identity is preserved from one end of a layer

connection to the other.

SEL

Selector: A subfield carried in SETUP message part of ATM endpoint address Domain specific Part (DSP) defined by ISO 10589, not used for ATM network routing, used by ATM end systems only.

SRTS

Synchronous residual Time Stamp: A clock recovery technique in which difference signals between source timing and a network reference timing signal are transmitted to allow reconstruction of the source timing at the destination.

SSCOP

Service Specific Connection Oriented Protocol: An adaptation layer protocol defined in ITU-T Specification: Q.2110.

SSCS

Service Specific Convergence Sublayer: The portion of the convergence sublayer that is dependent upon the type of traffic that is being converted.

SVC

Switched Virtual Circuit: A connection established via signaling. The user defines the endpoints when the call is initiated.

SVE

SAP Vector Element: The SAP address may be expressed as a vector, (ATM_addr, ATM_selector, BLLI_id2, BLLI_id3, BHLI_id), where:

- ATM_addr corresponds to the 19 most significant octets of a device's 20-octet ATM address (private ATM address structure) or the entire E.164 address (E.164 address structure)
- ATM_selector corresponds to the least significant octet of a device's 20-octet ATM address (private ATM address structure only)
- BLLI_id2 corresponds to an octet in the Q.2931 BLLI information element that identifies a layer 2 protocol corresponds to a set of octets in the Q.2931 BLLI information element that identify a layer 3 protocol
- BLLI_id3
- BHLI_id corresponds to a set of octets in the Q.2931 BHLI information element that identify an application (or session layer protocol of an application)

Each element of the SAP vector is called a SAP Vector Element, or SVE. Each SVE consists of a tag, length, and value field.

UNI

User-Network Interface: An interface point between ATM end users and a private ATM switch, or between a private ATM switch and the public carrier ATM network; defined by physical and protocol specifications per ATM Forum UNI documents. The standard adopted by the ATM Forum to define connections between users or end stations and a local switch.

UPC

Usage Parameter Control: Usage Parameter Control is defined as the set of actions taken by the network to monitor and control traffic, in terms of traffic offered and validity of the ATM connection, at the end-system access. Its main purpose is to protect network resources from malicious as well as unintentional misbehavior, which can affect the QoS of other already established connections, by detecting violations of negotiated parameters and taking appropriate actions.

VBR

Variable Bit Rate: An ATM Forum defined service category which supports variable bit rate data traffic with average and peak traffic parameters.

VCI

Virtual Channel Identifier: A unique numerical tag as defined by a 16 bit field in the ATM cell header that identifies a virtual channel, over which the cell is to travel.

VPI
: Virtual Path Identifier: An eight bit field in the ATM cell header which indicates the virtual path over which the cell should be routed.

VTOA
: Voice and Telephony Over ATM: The ATM Forum voice and telephony over ATM service interoperability specifications address three applications for carrying voice over ATM networks; desktop (or LAN services), trunking (or WAN services), and mobile services.

1.2.2 Data Link Provider Interface (DLPI) Glossary

Called
: The DLS user in connection mode that processes requests for connections from other DLS users.

Calling
: The DLS user in connection mode that initiates the establishment of a data link connection.

Communication endpoint
: The local communication channel between a DLS user and DLS provider.

Connection establishment
: The phase in connection mode that enables two DLS users to create a data link connection between them.

Connectionless mode
: A mode of transfer in which data is passed from one user to another in self-contained units with no logical relationship required among the units.

Connection management stream
: A special stream that will receive all incoming connect indications destined for DLSAP addresses that are not bound to any other streams associated with a particular PPA.

Connection mode
: A circuit-oriented mode of transfer in which data is passed from one user to another over an established connection in a sequenced manner.

Connection release
: The phase in connection mode that terminates a previously established data link connection.

Data link service data unit
: A grouping of DLS user data whose boundaries are preserved from one end of a data link connection to the other.

Data transfer
: The phase in connection and connectionless modes that supports the transfer of data between two DLS users.

DLS
: Data Link Service.

DLSAP
: A point at which a DLS user attaches itself to a DLS provider, to access data link services.

DLSAP address
: An identifier used to differentiate and locate specific DLS user access points to a DLS provider.

DLS provider
: The data link layer protocol that provides the services of the Data Link Provider Interface.

DLSDU
: Data Link Service Data Unit.

DLS user
: The user-level application or user-level or kernel-level protocol which accesses the services of the data link layer.

ISO
: International Organization for Standardization.

Local management
: The phase in connection and connectionless modes in which a DLS user initializes a stream and binds a DLSAP to the stream. Primitives in this phase generate local operations only.

OSI
: Open Systems Interconnection.

PPA
: Physical Point of Attachment: the point at which a system attaches itself to a physical communications medium.

PPA identifier
: An identifier of a particular physical medium over which communication transpires.

QOS
: Quality of service: characteristics of transmission quality between two DLS users.

2. Proposed Architecture

The proposed architecture is as follows :

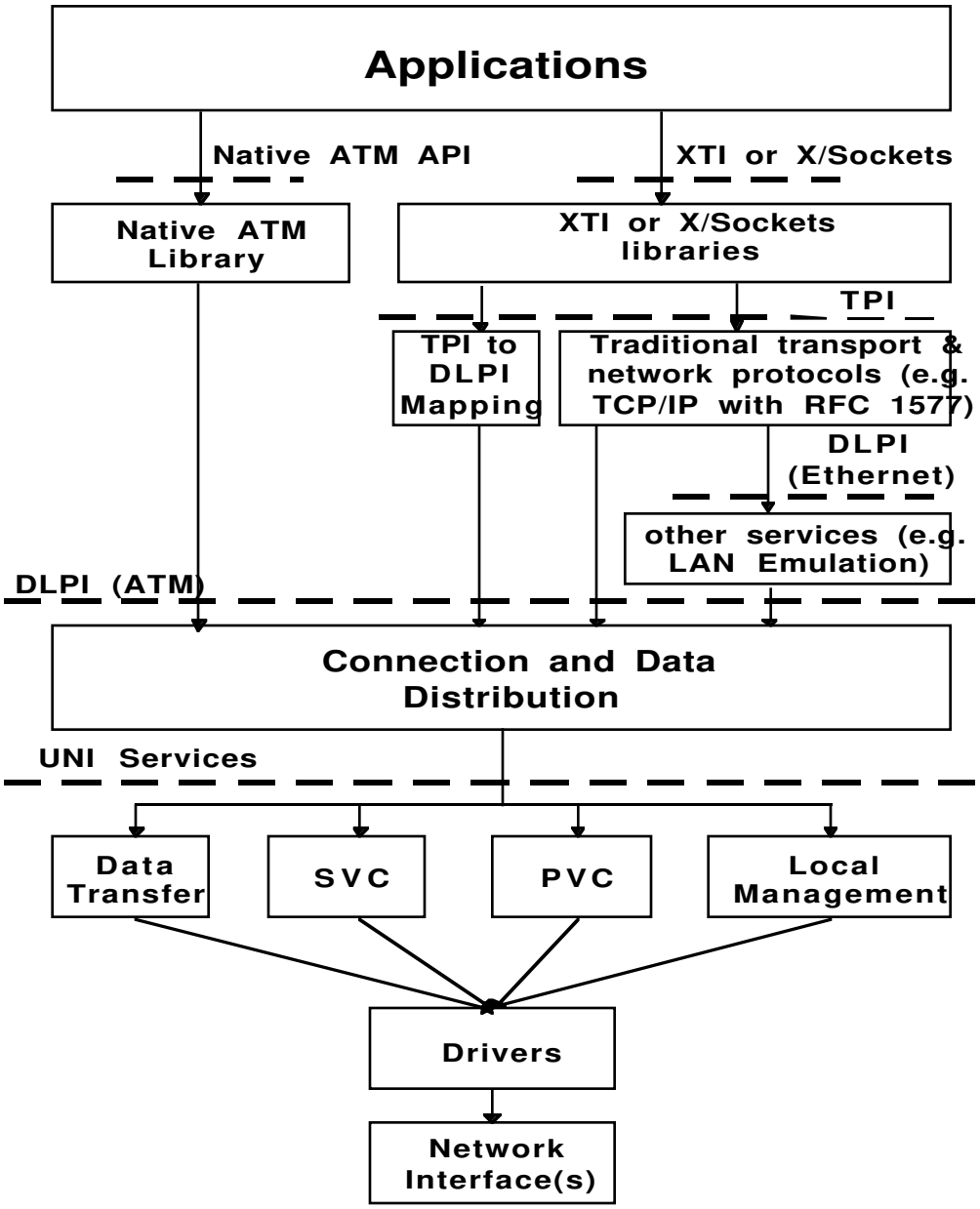


Figure 1: ATM Architecture

Note: While DLPI can be used as the interface to access Native ATM Services, it is by no means the only way to perform implementations.

3. ATM Provider DLPI Addendum

3.1 Native ATM Services Data Link Provider Specific Addendum

This chapter fulfills the requirements of DLPI Annex E which requires certain information about a DLS Provider.

3.1.1 DLSAP Address space

The ATM DLS provider doesn't support the ability to dynamically allocate DLSAPs other than the requested DLSAP in a DL_BIND_REQ.

An ATM DLS Provider can optionally support a “connection management” stream, via a bind with all tags (to be defined in next section) set to ANY. If the ATM DLS Provider does not support a “connection management” stream, then binds with all tags set to ANY will be rejected with DL_UNSUPPORTED.

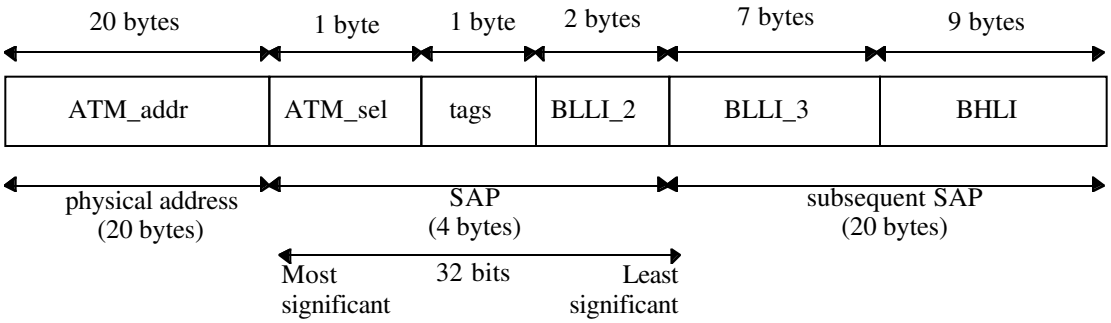
3.1.1.1 DLSAP format

Specification of the DLSAP address space is based upon “Native ATM Services : Semantic Description v1.0” from the ATM Forum. Following this specification, a DLSAP is made of 5 SAP Vector Elements (SVEs) :

- ATM_addr SVE is realized by “ATM_addr tag”, “ATM_addr type” plus “ATM_addr value” (note that ATM_addr type concatenated with ATM_addr value contains the information defined by the SVE_value field of ATM_addr SVE in “Native ATM Services: Semantic Description”),
- ATM_selector SVE is realized by “ATM_selector tag” plus “ATM_selector value”,
- BLLI_id2 SVE is realized by “BLLI_id2 tag” plus “BLLI_id2 value”,
- BLLI_id3 SVE is realized by “BLLI_id3 tag” plus “BLLI_id3 value”
- and BHLI_id SVE is realized by “BHLI_id tag” plus “BHLI_id value”.

Following DLPI specification, the DLSAP can be cut into 3 parts :

- the physical address,
- the SAP provided with DL_BIND_REQ
- and the subsequent SAP provided with DL_SUBS_BIND_REQ (Hierarchical).



The physical address part contains the ATM_addr type and value. The SAP part contains the ATM_selector and BLLI_id2 values plus all tags. The subsequent SAP part contains the BLLI_id3 and BHLI_id values. This results in a 40 bytes long DLSAP.

3.1.1.1.1 SVE tagging

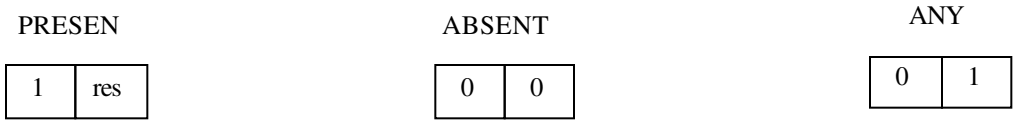
Each SVE of an ATM SAP address may be tagged as PRESENT, ABSENT or ANY.

If tagged as PRESENT, the SVE contains valid data, and incoming connections with signalling Information Element (IE) matching this SVE are routed to the bound DLS user. SVE value will also be included in signalling IE of outgoing connections.

If tagged as ABSENT, the SVE contains invalid data, and incoming connections with no signalling IE corresponding to this SVE are routed to the bound DLS user. There will be no signalling IE corresponding to this SVE in an outgoing connection.

If tagged as ANY, the SVE contains invalid data, and incoming connections routed to the bound DLS user may or not contain a signalling IE corresponding to this SVE. There is no signalling IE corresponding to this SVE in an outgoing connection.

All SVE tags are coded on at most 2 bits as following :



3.1.1.1.2 SVE encoding

The encoding chosen for the 5 SVEs is the following :

3.1.1.1.2.1 ATM_addr

ATM_addr type and value: the 3 types of ATM_addr are the following:

NSAP

ATM_addr type: NSAP	1 byte
19 bytes of NSAP	19 bytes

E.164

ATM_addr type: E.164	1 byte
1 to 15 bytes of E.164 address (IA5) followed by '#'	n bytes (2 <= n <=16)
padding (3 to 17 bytes)	(19 - n) bytes

Note: the total size is 20 bytes

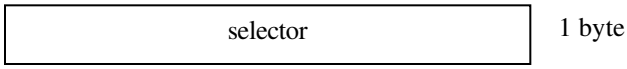
VPI/VCI : for PVCs

ATM_addr type: VPI/VCI	1 byte
vpi	1 bytes
vci	2 bytes
padding	16 bytes

For PVCs there is a need to convey the VPI and VCI instead of the true ATM Address. This format of ATM_addr value is only used in conjunction with the extended BLLI_id2 value.

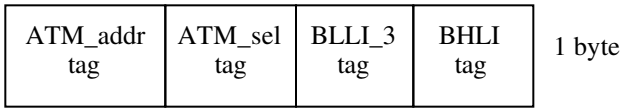
3.1.1.1.2.2 ATM_selector

ATM_selector value



3.1.1.1.2.3 Tagging

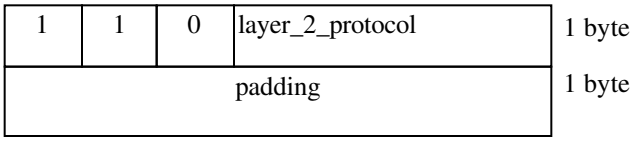
Tags (except the BLLI_id2 tag)



This field is used to encode the ATM_addr, ATM_selector, BLLI_id3 and BHLI_id tags. BLLI_id2 tag is encoded in the first 2 bits of the next field. Thus, all SVE tags are present in the SAP part and provided with the DL_BIND_REQ primitive.

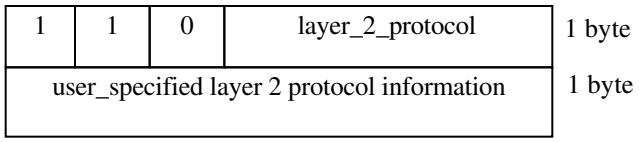
3.1.1.1.2.4 BLLI_id2

BLLI_id2 tag and value (option 1)



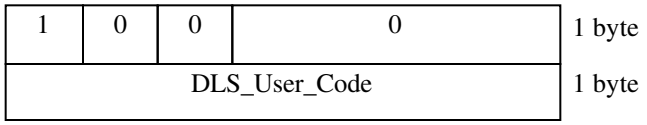
This option is only supported with a value for layer_2_protocol different from “User Specified” (0x10).

BLLI_id2 tag and value (option 2)



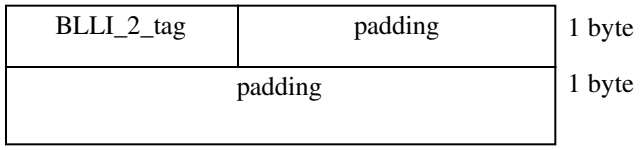
This option is only supported with a value for layer_2_protocol equal to "User Specified" (0x10).

BLLI_id2 tag and value (when used for PVCs)



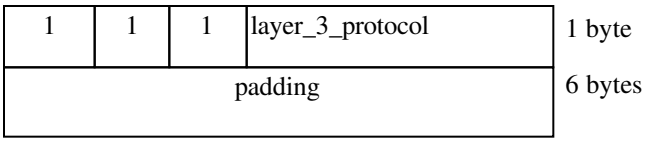
When the Virtual Circuit is a PVC, then this field is used to identify the DLS User. The DLS User Code may range between 0x00 and 0xFF with values reserved for Signaling (0xFF), ILMI (0xFE), Classical IP (0xFD), and Direct Access (0x00). The range 0x01-0xFC is reserved.

BLLI_id2 value (when BLLI_id2 tag equals ABSENT or ANY)



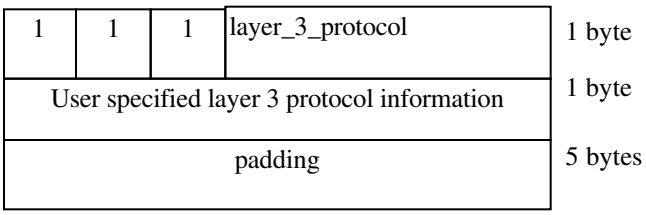
3.1.1.1.2.5 BLLI_id3

BLLI_id3 value (option 1)



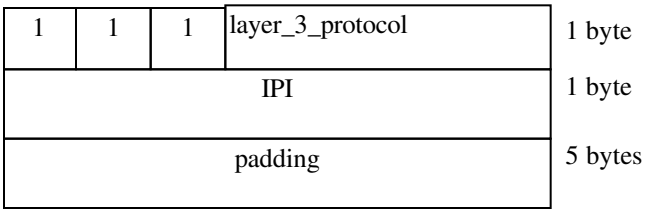
This option is only supported with layer_3_protocol not equal to ISO_IEC_TR_9577 (0x0B) or User Specified (0x20).

BLLI_id3 value (option 2)



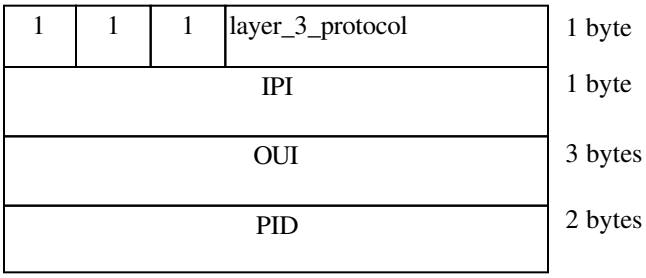
This option is only supported with layer_3_protocol equal to User Specified (0x20).

BLLI_id3 value (option 3)



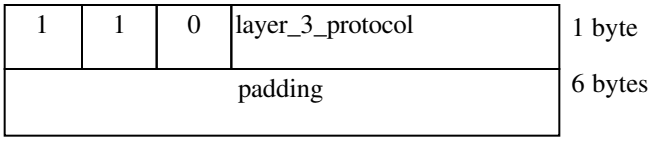
This option is only supported with layer_3_protocol equal to ISO_IEC_TR_9577 (0x0B) and IPI is not equal to IEEE_802_1_SNAP (0x80).

BLLI_id3 value (option 4)



This option is only supported with layer_3_protocol equal to ISO_IEC_TR_9577 (0x0B) and IPI equal to IEEE_802_1_SNAP (0x80).

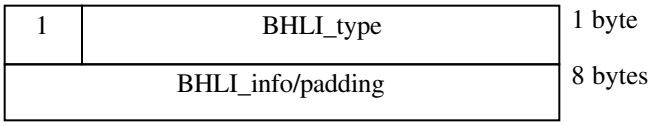
BLLI_id3 value (option 5)



This option is only supported with layer_3_protocol equal to ISO_IEC_TR_9577 (0x0B) and no International Protocol Identifier (IPI) is specified. For more information on this option, see Note 2 on page 216 of the UNI spec. “User-Network Interface specification version 3.1”.

3.1.1.1.2.6 BHLI_id

BHLI_id value (options 1, 2, 3)



For option 1, BHLI_type is equal to ISO (0x00) and the 8 bytes of BHLI_info are used.

For option 2, BHLI_type is equal to User_Specific (0x01) and the 8 bytes of BHLI_info are used.

For option 3, BHLI type is equal to Vendor_Specific (0x03) and only the first 7 bytes of BHLI_info are used.

3.1.1.2 DLSAP registration

When providing the SAP part of the DLSAP with DL_BIND_REQ, if BLLI_id3 or BHLI_id SVEs are tagged as present, a DL_SUBS_BIND_REQ (Hierarchical) is necessary to provide a subsequent SAP.

No DL_CONNECT_IND will be received by the DLS user until this DL_SUBS_BIND_REQ (Hierarchical) is performed. A DL_CONNECT_REQ sent by the DLS user before the DL_SUBS_BIND_REQ will generate an outgoing call as if BLLI_id3 and BHLI_id SVEs were tagged as ABSENT.

If BLLI_id3 and BHLI_id SVEs are tagged as ABSENT or ANY in SAP part, no DL_SUBS_BIND_REQ (Hierarchical) is required. A DL_SUBS_BIND_REQ (Hierarchical) would be rejected by a DL_ERROR_ACK.

Multiple ATM Address support: One port can have several ATM addresses. When using the straight bind method, the ATM_addr SVE corresponds to the "primary" ATM address of the port (Available after the DL_ATTACH_REQ / DL_OK_ACK in a DL_INFO_ACK). If the DLS user wants to use another ATM Address, it must bind with the ATM_addr tag set to ABSENT and issue a DL_SUBS_BIND_REQ (Peer) with the ATM_addr value set to a port "secondary" ATM Address.

A DL_BIND_REQ (***dl_max_conind*** > 0) with an ATM_addr tag set to ANY means the DLS user expects to receive incoming calls with Information Elements (IEs) matching the other SVEs but regardless of the called address IE.

Multiple BLLI support for outgoing calls: Three DLSAPs can be bound to a stream using bind, Hierarchical bind and Peer subsequent binds to send a setup with up to three BLLIs. The BLLIs are sent in the order of the DLSAP binds. The dl_qos BLLI entries containing BLLI attributes must be in the same order as the bound BLLIs.

Note: the ATM_addr, ATM_selector and ATM_BHLI_id tags must be set to ABSENT in the second and third DLSAP bound.

Multiple BLLI support for incoming calls: At least three DLSAPs can be bound to a stream using bind, Hierarchical binds and Peer subsequent binds to accept a setup with multiple BLLIs. See the Incoming Call Routing below for further details.

No peer subsequent bind is supported outside the two above cases: Multiple ATM Address support (one peer bind), Multiple BLLI support (two or three peer binds with the initial bind ***dl_max_conind*** set to 0).

PVC support: the ATM_addr tag must be set to ABSENT, the ATM_selector and BLLI_id2 values must be set, the BLLI_id3 and BHLI_id tags must be set to ABSENT. DL_BIND_ACK returns the DLSAP with the same tagging and values.

DLSAP registration

After DL_BIND_REQ and DL_SUBS_BIND_REQ, the full DLSAP is registered if it doesn't match with an already registered DLSAP. A DLSAP matches with a registered DLSAP if at least one of its SVEs matches with a SVE of the registered DLSAP according to the following rules :

Table 1: matching rules for DLSAP registration

new SVE \ registered SVE	ABSENT	PRESENT	ANY
ABSENT	match	no match	match
PRESENT	no match	match if SVE values match	match
ANY	match	match	match

These rules prevents SAP overlapping, i.e. for the distribution of the incoming call notification to the appropriate application, an incoming call can match with at most one registered DLSAP.

Incoming Call Routing

On an incoming setup, a destination DLSAP is calculated from setup information elements. If this destination DLSAP matches with a registered DLSAP, the incoming setup is indicated by DL_CONNECT_IND to the DLS user bound to the registered DLSAP. If the destination DLSAP does not match with any of the registered DLSAP, the incoming setup is rejected. A destination DLSAP matches with a registered DLSAP if all its SVEs match the SVEs of a registered DLSAP according to the following rules :

Table 2: matching rules for incoming call routing

destination SVE\registered SVE	ABSENT	PRESENT	ANY
ABSENT	match	no match	match
PRESENT	no match	match if SVE values match	match

As SAP overlapping is not allowed, only one DLS user will be indicated in the incoming call.

Note: if several BLLIs are present in the setup, up to three DLSAPs are calculated and the first one matching with a registered DLSAP is used as the called DLSAP in the DL_CONNECT_IND. If several DLSAPs are bound to the stream which match the BLLIs, those are included in the dl_qos structure along with the retained DLSAP which is pointed to by the BLLI_Selector field and included in the dl_called_addr. The BLLIs not matching a bound DLSAP of the stream are not included in the dl_qos. The order in the dl_qos is the same as the BLLIs in the setup. In the DL_CONNECT_RES, the BLLI_Selector is used to select an alternate BLLI if the one returned in the DL_CONNECT_IND does not suit the application need. However the retained BLLI must be one of those presented in the DL_CONNECT_IND dl_qos structure.

- Example:

If three dlsaps are bound with a stream (and in this order):
BLLI_id2 any, BLLI_id3 SNAP OUI 0x000000 PID 0x0020
BLLI_id2 any, BLLI_id3 SNAP OUI 0x000000 PID 0x0030
BLLI_id2 any, BLLI_id3 SNAP OUI 0x000000 PID 0x0040

A setup comes in with:
BLLI_id2 any, BLLI_id3 SNAP OUI 0x000000 PID 0x0040
BLLI_id2 any, BLLI_id3 SNAP OUI 0x000000 PID 0x0030
BLLI_id2 any, BLLI_id3 SNAP OUI 0x000000 PID 0x0090

The DL_CONNECT_IND is made of:
Called DLSAP BLLI_id3: IEEE 802.1 SNAP OUI 0x000000 PID 0x0040
dl_qos .BLLI_Parameters[0] .BLLI_id3_value SNAP OUI 0x000000 PID 0x0040
dl_qos .BLLI_Parameters[1] .BLLI_id3_value SNAP OUI 0x000000 PID 0x0030
dl_qos .BLLI_Selector = 0; /* Matches the DL_CONNECT_IND called dlsap */

The DL_CONNECT_RES can be:
dl_qos .BLLI_Parameters[0] .BLLI_id3_value SNAP OUI 0x000000 PID 0x0040
dl_qos .BLLI_Parameters[1] .BLLI_id3_value SNAP OUI 0x000000 PID 0x0030

dl_qos .BLLI_Selector = 1; /* not happy with PID 0x40, use 0x30 instead */

This causes the connect to be sent with:
BLLI_id3: SNAP OUI 0x000000 PID 0x0030

DLSAP values in DL_CONNECT_REQ

The called DLSAP in DL_CONNECT_REQ must follow a number of restrictions. All the SVEs must be tagged as PRESENT or ABSENT. The ATM_addr and ATM_selector SVEs must be tagged as PRESENT. The tagging and values of SVEs contained in SAP and subsequent SAP parts of called DLSAP must be consistent with DLSAP registered in DL_BIND_REQ, DL_SUBS_BIND_REQ, i.e.:

for SVCs: called blli_2, blli3 and bhli SVE must be equal to those of one of the bound DLSAPs, and the selector and address must be present,

for PVCs: extended blli_2 must be equal, selector and VPI/VCI must be present.

For SVCs, the SETUP PDU will contain IEs indicated by the SVE tagging in called DLSAP. If several DLSAP were bound to the stream, several BLLIs are included in the SETUP PDU (See DLSAP Registration, Multiple BLLI support).

For PVCs, the ATM_addr value will contain the VPI/VCI information enabling stream attachment to PVC. The DL_CONNECT_REQ may cause the PVC to be created according to parameters passed in the DL_QoS. The main operation process is similar to the one described for SVCs. The PVC does not exist anymore after the disconnection when the DLS User has sent a DL_DISCONNECT_REQ, or after the DLS User has received a DL_DISCONNECT_IND when the port operating status becomes OutOfService (in which case the DLS User has to re-create the PVC, as it would for an SVC).

However, as there is no signalling for PVCs, the success of the open of the PVC will depend on existing conditions.

- If the PVC does not exist, then it can be created using the information in the DL_QoS, and attached to the DLS User stream.

- If the PVC exists, it has already been configured by a mean outside of the scope of DLPI Addendum for ATM, then:

  - If the DL_QoS is not empty (DL_qos field size <> 0 and some parameters tagged as VALID or DEFAULT), the DL_CONNECT_REQ is refused.
  - if the DL_QoS is empty (DL_qos field size = 0 or all parameters tagged as INVALID), the DL_CONNECT_REQ is processed:
    - if the PVC is in use, the DL_CONNECT_REQ is rejected, the PVC cannot be shared by more than one DLS User stream.

- if the PVC is not in use (case where the PVC had been pre-configured previously), the PVC is attached to the DLS User stream.

Information necessary in the DL_QoS for a PVC opened as an SVC

- ADL_AAL_Parameters_t AAL_Parameters;
  This field is used.
- ADL_ATM_Traffic_Descriptor_t ATM_Traffic_Descriptor;
  This field is used.
- ADL_Broadband_Bearer_Capability_t Broadband_Bearer_Capability;
  This field may be used to specify Point to Multipoint capability for the PVC.
- ADL_BLLI_Parameters_t BLLI_Parameters[3];
  This field is not used.
- ADL_Subaddress_t Called_Party_Subaddress;
  This field is not used.
- ADL_Calling_Party_Number_t Calling_Party_Number;
  This field is not used.
- ADL_Subaddress_t Calling_Party_Subaddress;
  This field is not used.
- ADL_QOS_t QOS;
  This field may be used. But its purpose of providing criteria for routing according to QoS parameters will be useless.
- ADL_Transit_Network_Selection_t Transit_Network_Selection;
  This field is not used, as it is used for the routing inside the ATM network.

For PVCs configured by means beyond the scope of the DLPI Addendum for ATM, see Note below on page **Note**22.

DLSAP values in DL_CONNECT_IND

For SVCs, as the Calling Party Number IE is optional in the SETUP message, the DL_CONNECT_IND primitive may not have a calling DLSAP part. The dl_calling_addr_length will then be equal to 0. The called DLSAP part of DL_CONNECT_IND will contain the destination DLSAP extracted from incoming setup parameters, with all SVEs only tagged as PRESENT or ABSENT.

For PVCs, the DL_CONNECT_IND will contain the calling and called DLSAP parts. The Extended ATM_addr value in the called DLSAP part of the DL_CONNECT_IND will be

tagged as ABSENT. The Extended ATM_addr value in the calling DLSAP part of the DL_CONNECT_IND will contain the VPI/VCI information.

Note: For PVCs configured by means beyond the scope of this DLPI Addendum for ATM, the DLS User will receive DL_CONNECT_IND primitives if the port operating status is InService and PVCs are configured or when PVCs are configured and the port operating status becomes InService. The DLS User will receive a DL_DISCONNECT_IND when the PVC is unconfigured or when the port operating status becomes OutOfService. This is a convenient way to synchronize the PVC connections with the state of the card and PVC configuration. For those PVCs, the DL_CONNECT_XYZ primitives must be thought as a way to link an application with the connection rather than establishing the connection over the network.

3.1.2 PPA Access and Control

The PPA number which is passed in the DL_ATTACH_REQ primitive should correspond to the ATM port the DLS User is being attached to.

The PPA Initialization method is the pre-initialization. There is no PPA de-initialization. The pre-initialization and de-initialization are performed using management tools which are outside the scope of this specification.

When PPAs are pre-initialized, DL_CONNECT_REQ can only be successful (SVC DLSAPs) after the ATM network prefix is configured. This ATM network prefix can be obtained using ILMI (See UNI 3.0/1) or specified as part of the pre-initialization. At de-initialization time all connections are released (DL_DISCONNECT_IND). If the ATM network prefix changes, the SVC connections are released (DL_DISCONNECT_IND).

If there are several ATM Addresses attached to the port (to the PPA), only one called “primary” ATM Address is returned in the DL_INFO_ACK. The other addresses are available for use if a peer subsequent bind has been issued (See DLSAP Registration, Multiple ATM Addresses support).

If the operating status of the port is OutOfService, all connections are released. When the operating status of the interface becomes either InService or LoopBack, the configured PVCs are checked against the bound saps and if there is a match DL_CONNECT_IND messages are sent upstream.

3.1.3 Quality of Service

The Quality of Service corresponds to the Connection Attributes of the Native ATM Services Connection Attributes minus the information already included in the DLSAP. The values of the QoS fields follow the constraints specified in the ATM UNI 3.0 (3.1) and in the Native ATM Services (Version 1.0).

The Appendix A “ATM Header File” on page 34 defines the necessary data.

3.1.4 DL_INFO_ACK Values

dl_max_sdu	65535 (AAL-5 Max SDU size).
dl_min_sdu	1.
dl_addr_length	40 (ADL_DLSAP_LEN) or 0 (No Address Registration yet, or not attached).
dl_mac_type	DL_ATM.
dl_sap_length	0 no SAP bound, -4 BLLI_id2 bind, -20 BLLI_id3 and/or BHLI subsequent bind.
dl_service_mode	DL_CODLS.
dl_qos_length	sizeof(ADL_dl_qos) if connected, 0 if not connected.
dl_qos_range_length	0. The DLS Provider does not return the range of QoS values.
dl_provider_style	DL_STYLE2.
dl_addr_offset	If the stream is bound, then ***dl_addr_length*** is not zero, and the full DLSAP is returned with the SVE tags set depending on the subsequent bind.
dl_version	DLPI 2.0.0 version (Whatever constant this is).
dl_brdcst_addr_length	0.
dl_brdcst_add_offset	0.

3.1.5 Supported Services

ATM DLPI provides a connection-mode oriented style 2 DLPI.

The primitives supported are:

- DL_INFO_REQ, DL_INFO_ACK
- DL_OK_ACK, DL_ERROR_ACK
- DL_BIND_REQ, DL_BIND_ACK, DL_UNBIND_REQ
- DL_SUBS_BIND_REQ, DL_SUBS_BIND_ACK, DL_SUBS_UNBIND_REQ
- DL_CONNECT_REQ, DL_CONNECT_IND,
- DL_CONNECT_RES, DL_CONNECT_CON
- DL_DISCONNECT_REQ, DL_DISCONNECT_IND
- DL_TOKEN_REQ, DL_TOKEN_ACK
- DL_DATA_REQ, DL_DATA_IND

The primitives not supported are:

- DL_RESET_REQ, DL_RESET_IND, DL_RESET_RES, DL_RESET_CON
- DL_ENABMULTI, DL_DISABMULTI
- DL_PROMISCON_REQ, DL_PROMISCOFF_REQ
- DL_XID_REQ, DL_XID_IND, DL_XID_RES, DL_XID_CON
- DL_TEST_REQ, DL_TEST_IND, DL_TEST_RES, DL_TEST_CON
- All primitives related to Connectionless-mode or acknowledged Connectionless-mode Data transfer.

4. Mapping of Primitives

4.1 Mapping of Native ATM Services primitives to ATM DLPI

4.1.1 State mapping

Table 3: State mapping

ATM API state	DLPI state	Comments
A0	-	No stream opened
A1	UNATTACHED	stream is opened, using **open**
A2	IDLE	Provided the bind ***dl_max_conind*** == 0
A3	OUTCON_PENDING	
A4	IDLE	Provided the bind ***dl_max_conind*** > 0 but a Subsequent bind is required
A5	IDLE	Provided the bind ***dl_max_conind*** > 0 and a Subsequent bind is not required or already done
A6	INCON_PENDING	
A7	CONN_RES_PENDING	
A8	DATAXFER	
A9	**Not Supported**	This feature requires an extension to DLPI 2.0.0
A10	DATAXFER	dl_qos in DL_CONNECT_IND specified a point to multipoint connection.
A11	-	stream has been closed, using **close**

4.1.2 Primitives mapping table

Table 4: Primitives mapping

Native ATM Services : Semantic Description	DLPI ATM Addendum
ATM_abort_connection request	**DL_DISCONNECT_REQ**
ATM_accept_incoming_call response	**DL_CONNECT_RES**
ATM_add_party request	**Not Supported**, requires an extension to DLPI 2.0.0.
ATM_associate_endpoint request	**open** (open a file descriptor or a stream).
ATM_add_party_reject confirm	**Not Supported**, requires an extension to DLPI 2.0.0.
ATM_add_party_success confirm	**Not Supported**, requires an extension to DLPI 2.0.0.
ATM_arrival_of_incoming_call indication	**DL_CONNECT_IND**
ATM_call_release request	**DL_DISCONNECT_REQ** followed by a **close**

ATM_call_release indication	**DL_DISCONNECT_IND** followed by a **close**
ATM_connect_outgoing_call request	**DL_CONNECT_REQ**
ATM_drop_party request	**Not Supported**, requires an extension to DLPI 2.0.0.
ATM_drop_party indication	**Not Supported**, requires an extension to DLPI 2.0.0.
ATM_get_local_port_info request	**DL_ATTACH_REQ / DL_OK_ACK, DL_INFO_REQ / DL_INFO_ACK**
ATM_P2MP_call_active confirm	**Incoming Call (state A7 to A10)** There is no mapping because **DL_OK_ACK** is locally generated. Note that transmitting data immediately after **DL_OK_ACK** is not guaranteed to work, because the virtual circuit may not be setup. This a common ATM problem. The DLS Provider may optionally buffer data during this transitory period. **Outgoing Call (state A3 to A9)** **Not Supported,** requires an extension to DLPI 2.0.0.
ATM_P2P_call_active confirm	**Outgoing Call (state A3 to A8)** **DL_CONNECT_CON** **Incoming Call (state A7 to A8)** There is no mapping because **DL_OK_ACK** is locally generated. Note that transmitting data immediately after **DL_OK_ACK** is not guaranteed to work, because the virtual circuit may not be setup. This a common ATM problem. The DLS Provider may optionally buffer data during this transitory period.
ATM_prepare_incoming_call request	**DL_ATTACH_REQ**, **DL_OK_ACK**, **DL_BIND_REQ / DL_BIND_ACK**, [**DL_SUBS_BIND_REQ / DL_SUBS_BIND_ACK**]
ATM_prepare_outgoing_call request	There is no direct mapping of this primitive. Instead the setup of the Connection Attributes which can be done in A2 is done using: **DL_ATTACH_REQ**, **DL_OK_ACK**, **DL_BIND_REQ / DL_BIND_ACK**, [**DL_SUBS_BIND_REQ / DL_SUBS_BIND_ACK**]
ATM_query_connection_attributes request	Most of the connection attributes are available using **DL_INFO_REQ / DL_INFO_ACK**. Other connection attributes are either provided by the DLS user or returned at bind or connection time by the

	DLS provider.
ATM_reject_incoming_call response	**DL_DISCONNECT_REQ** (may be followed by a close)
ATM_set_connection_attributes request	There is no mapping. However all the connection attributes which can be set when the state is either A2 or A6 can be set by providing the attribute value in DLPI primitives exchanged while leaving one of those states: (**DL_CONNECT_REQ / DL_CONNECT_RES**).
ATM_wait_on_incoming_call request	There is no direct mapping, see **ATM_prepare_incoming_call**.
ATM_send_data request	**DL_DATA_REQ**
ATM_receive_data indication	**DL_DATA_IND**

4.1.3 Primitives parameters general mapping

endpoint_identifier : the stream, file descriptor in user space, read or write queue in kernel space.

ATM_abort_connection request (*endpoint_identifier*, *cause*)

DL_DISCONNECT_REQ (state A3, A8, A10) followed by a **close**

dl_reason is only a subset of the ***cause***

close (state A1, A2, A3, A4, A5, A6, A7, A8, A9, A10)

Note: VTOA needs to use the cause code DL_CONREJ_PERMANENT_COND in order to be able to retry connection attempt using another AAL type.

ATM_accept_incoming_call response (*endpoint_identifier*, *new_endpoint_identifier*)

DL_CONNECT_RES

dl_resp_token corresponds to ***new_endpoint_identifier***

dl_qos_length, ***dl_qos_offset*** corresponds to the ***Connection Attributes*** which can be set while in state A6.

ATM_add_party request (endpoint_identifier, leaf_identifier, leaf_ATM_address)

Not Supported, would require an extension to DLPI 2.0.0.

ATM_add_party_reject confirm (endpoint_identifier, leaf_identifier, rejection_cause)

Not Supported, would require an extension to DLPI 2.0.0.

ATM_add_party_success confirm (endpoint_identifier, leaf_identifier)

Not Supported, would require an extension to DLPI 2.0.0.

ATM_arrival_of_incoming_call indication (endpoint_identifier)

DL_CONNECT_IND

dl_called_addr, ***dl_calling_addr***, ***dl_qos*** hold Connection Attributes available while in state A6.

ATM_associate_endpoint request (endpoint_identifier)

open (open a file descriptor or a stream).

ATM_call_release request (endpoint_identifier, release_cause)

DL_DISCONNECT_REQ followed by a **close**

dl_reason is only a subset of the ***release_cause***

ATM_call_release indication (endpoint_identifier, release_cause)

DL_DISCONNECT_IND followed by a **close**

dl_reason is only a subset of the ***release_cause***

ATM_connect_outgoing_call request (endpoint_identifier, destination_SAP)

DL_CONNECT_REQ

dl_dest_addr corresponds to the ***destination_SAP***

dl_qos corresponds to the Connection Attributes which can be modified while in state A2

ATM_drop_party request (endpoint_identifier, leaf_identifier, drop_cause)

Not Supported, would require an extension to DLPI 2.0.0.

ATM_drop_party indication (endpoint_identifier, leaf_identifier, drop_cause)

Not Supported, would require an extension to DLPI 2.0.0.

ATM_get_local_port_info request (port_number, address_list, line_rate)

DL_ATTACH_REQ / DL_OK_ACK, DL_INFO_REQ / DL_INFO_ACK

dl_ppa (DL_ATTACH_REQ) corresponds to the port_number

dl_qos_range contains the default ***FWD_PCR_CLP0*** with ***BEST_EFFORT*** set and therefore should advertise the ***line_rate*** in cells/s

dl_addr corresponds to one of the ATM addresses for the ***dl_ppa***.

Note: the complete ***address_list*** is not available using **DL_INFO_ACK**.

Retrieving the complete list of ATM addresses is a function belonging to the management plane which is out of the scope of DLPI.

ATM_P2MP_call_active confirm (endpoint_identifier)

DL_OK_ACK (DL_CONNECT_RES) with the ***dl_qos*** in DL_CONNECT_IND stating a point to multipoint connection (state A7 to A10).

Not Supported (state A3 to A9), would require an extension to DLPI 2.0.0.

ATM_P2P_call_active confirm (endpoint_identifier)

DL_OK_ACK (DL_CONNECT_RES) with the ***dl_qos*** in DL_CONNECT_IND stating a point to point connection (state A7 to A8).

DL_CONNECT_CON (state A3 to A8).

ATM_prepare_incoming_call request (endpoint_identifier, local_SAP, queue_size)

DL_ATTACH_REQ, **DL_OK_ACK**, **DL_BIND_REQ / DL_BIND_ACK**, **DL_SUBS_BIND_REQ / DL_SUBS_BIND_ACK**

dl_max_conind (DL_BIND_REQ) corresponds to ***queue_size***

dl_ppa (DL_ATTACH_REQ) is used to specify the ***port_number***

dl_sap (DL_BIND_REQ with ATM_addr tag not set to absent) is used to specify the ATM_selector SVE as well as the BLLI_id2 SVE of the ***local_SAP*** (With the primary ATM_Addr SVE of the ***dl_ppa***)

dl_subs_sap (DL_SUBS_BIND_REQ Hierarchical) is used to specify the BLLI_id3 SVE and / or BHLI_id SVE if any of the ***local_SAP*** (BLLI_id3 SVE tag or BHLI_id tag set to present).

dl_sap (DL_BIND_REQ with ATM_addr tag set to absent) is used to specify an ATM_addr SVE different than the port primary ATM_addr is to be used.

dl_subs_sap (DL_SUBS_BIND_REQ Peer) is used to specify the ***local_SAP***.

Note: the connection moves directly to A5 on the DL_BIND_REQ / DL_BIND_ACK if the ***local_SAP*** has been fully forwarded to the DLS provider.

ATM_prepare_outgoing_call request (endpoint_identifier)

There is no direct mapping of this primitive. Instead the setup of the Connection Attributes which can be done in A2 is done using:

DL_ATTACH_REQ, **DL_OK_ACK**, **DL_BIND_REQ / DL_BIND_ACK**, **DL_SUBS_BIND_REQ / DL_SUBS_BIND_ACK**

dl_ppa (DL_ATTACH_REQ) is used to specify the ***port_number***

dl_sap (DL_BIND_REQ with ATM_addr tag not set to absent) is used to specify the ATM_selector SVE as well as the BLLI_id2 SVE of the ***local_SAP*** (With the primary

ATM_Addr SVE of the ***dl_ppa***). This is the way the ***Calling Party Number*** and ***LAYER_2_ID*** connection attributes are set.

dl_subs_sap (DL_SUBS_BIND_REQ Hierarchical) is used to specify the BLLI_id3 SVE and / or BHLI_id SVE if any of the ***local_SAP*** (BLLI_id3 SVE tag or BHLI_id tag set to present). This is the way the ***LAYER_3_ID***, ***LAYER_3_USER_ID***, ***LAYER_3_IPI_ID***, ***LAYER_3_OUI_ID***, ***LAYER_3_PID_ID*** and ***BHLI*** connection attributes are set.

dl_sap (DL_BIND_REQ with ATM_addr tag set to absent) is used to specify an ATM_addr SVE different than the port primary ATM_addr is to be used.

dl_subs_sap (DL_SUBS_BIND_REQ Peer) is used to specify the ***Calling Party Number***, ***LAYER_2_ID***, ***LAYER_3_ID***, ***LAYER_3_USER_ID***, ***LAYER_3_IPI_ID***, ***LAYER_3_OUI_ID***, ***LAYER_3_PID_ID*** and ***BHLI*** connection attributes are set.

Note: the connection moves directly to A2 on the DL_BIND_REQ / DL_BIND_ACK if the ***local_SAP*** has been fully forwarded to the DLS provider.

Note: also DL_SUBS_BIND_REQ Peer can be used to specify more than one BLLI.

dl_max_conind (DL_BIND_REQ) is set to 0.

ATM_query_connection_attributes request (endpoint_identifier, attribute_name, attribute_value)

Most of the connection attributes are available using DL_INFO_REQ / DL_INFO_ACK. Other connection attributes are either provided by the DLS user or returned at bind or connection time by the DLS provider.

The following table specify the connection attributes not always available in a DL_INFO_ACK and how to obtain those.

Table 5:

Connection Attributes	
BLLI	the alternate BLLI values are available in DL_CONNECT_IND. On the DL_CONNECT_RES, only one BLLI is retained. The retained BLLI is available in DL_INFO_ACK.
Called Party Number	If the DLS user initiate the connection, the DLS user provides it at DL_CONNECT_REQ. If the DLS user accepted the connection, the attribute is available in the DL_INFO_ACK.
Calling Party Number	If the DLS user accepted the connection, the attribute is available in the DL_CONNECT_IND. If the DLS user initiate the connection, the attribute is available in the DL_INFO_ACK.
Cause	**Not Supported**, would require an extension to DLPI 2.0.0.

ATM_reject_incoming_call response(*endpoint_identifier*, *rejection_cause*)

DL_DISCONNECT_REQ (may be followed by a close)

dl_reason is only a subset of the ***rejection_cause.***

dl_correlation is set to the **DL_CONNECT_IND *dl_correlation.***

ATM_set_connection_attributes request (*endpoint_identifier*, *attribute_name*, *attribute_value*)

There is no mapping. However all the connection attributes which can be set when the state is either A2 or A6 can be set by providing the attribute value in DLPI primitives exchanged while in one of those states.

The only attributes which can not be set are the cause attributes.

ATM_wait_on_incoming_call request(*endpoint_identifier*)

There is no direct mapping, see **ATM_prepare_incoming_call**.

ATM_send_data request(*endpoint_identifier, data_source*, *sending_result*)

DL_DATA_REQ

ATM_receive_data indication(*endpoint_identifier, receive_data*)

DL_DATA_IND

Note: The Management Plane is out of the scope of DLPI. Some DLPI primitive may be used to implement part of the management plane. However DLPI purpose is to provide the Control and Data plane only. An other interface suitable for management plane access or more general (to support MIB instrumentation access) needs to be defined and used.

5. ATM Header File

<atm_dlpi.h>

```
/*
**-header-------------------------------------------------------------
**
** Interface : AtmDlpi (ADL)  $Revision: 1.8 $ $Date: 1997/09/05 15:44:46 $
**
** Purpose : This header file defines what is required to implement
**            the ATM Forum Native ATM Services: Semantic Description
**            Version 1.0 (document af-saa-0048.000) over DLPI.
**            This file is part of the Required Information for DLS
**            Provider-Specific Addenda as described in the DLPI Specification.
**
**            Please use the following references for further details:
**               - Native ATM Services: Semantic Description Version 1.0
**               - ATM User-Network Interface Specification (UNI 3.0 and 3.1)
**               - Data Link Provider Interface Specification (Revision 2.0.0)
**
**----------------------------------------------------------------------
*/


/*
**-------------------------------------------------------
**-------------------------------------------------------
** DLSAP Address Space
**-------------------------------------------------------
**-------------------------------------------------------
*/


/*
```

```
** ATM Address : NSAP
**
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |             Network Prefix                                    |
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |             End System Identifier (ESI)                       |
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |             Selector                                          |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Note the following defines the ATM NSAP used at signaling, ILMI
** and other ATM components level. The ATM NSAP is not as such a
** Native ATM Services object but is convenient in the implementation.
*/


# define ADL_NET_PREFIX_OFF    0
# define ADL_NET_PREFIX_SIZE   13


# define ADL_ESI_OFF           13
# define ADL_ESI_SIZE          6


# define ADL_SEL_OFF           19
# define ADL_SEL_SIZE          1


# define ADL_ATM_ADDR_SIZE     20


typedef uint8_t    ADL_atm_addr_t[ADL_ATM_ADDR_SIZE],
     *ADL_atm_addr_ptr_t;
```

```
/*
** Atm Address Value
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                   ATM_addr_type (NSAP)                        |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                                                               |
** +                 19 Bytes of NSAP address (no selector)        +
** +                                                               +
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Atm Address Value
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                   ATM_addr_type (E164)                        |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                                                               |
** +                                                               +
** +       1 to 15 bytes of E.164 address terminated with '#'      +
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                17 to 3 Bytes of padding                       |
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Atm Address Value
** (extension)
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                   ATM_addr_type (vpi_vci)                     |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

```
** |                              VPI                              |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                              VCI                              |
** +                                                               +
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |                16 Bytes of padding                            |
** |                                                               |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Note the BLLI_id2 extension is used when the ATM_addr extension is used
*/


# define ADL_ATM_ADDR_TYPE_OFF     0


/*
** ATM_addr_type values
*/
# define ADL_NSAP                  0x82
# define ADL_E164                  0x81
# define ADL_VPI_VCI               0xff


/*
** NSAP ATM_addr_type
*/
# define ADL_ATM_ADDR_PREFIX_OFF   1
# define ADL_ATM_ADDR_ESI_OFF      14


/*
** E.164 ATM_addr_type
*/
# define ADL_ATM_ADDR_E164_OFF     1
```

```
# define ADL_ATM_ADDR_E164_SIZE    15


/*
** vpi_vci ATM_addr_type
*/
# define ADL_ATM_ADDR_VPI_OFF      1
# define ADL_ATM_ADDR_VCI_OFF      2
# define ADL_ATM_ADDR_VCI_SIZE     2


# define ADL_ATM_ADDR_VAL_SIZE     20


typedef uint8_t    ADL_ATM_addr_val_t[ADL_ATM_ADDR_VAL_SIZE],
     *ADL_ATM_addr_val_ptr_t;


/*
** ATM selector Value
**
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | ATM_selector                                                  |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
*/
# define ADL_ATM_SELECTOR_SEL_OFF          0x00


# define ADL_ATM_SELECTOR_VAL_SIZE         1


typedef uint8_t    ADL_ATM_selector_val_t[ADL_ATM_SELECTOR_VAL_SIZE],
     *ADL_ATM_selector_val_ptr_t;


/*
** Tags (except the BLLI_id2 tag)
**
```

```
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | ATM_addr       | ATM_selector   | BLLI_id3       | BHLI_id        |
** |      tag       |      tag       |      tag       |      tag       |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Each SVE present in dlsap may be tagged as present,
** absent or any.
** Tagging of any of the SVE has been concentrated in sap part.
**
** Note when the primary sap is registered at bind
** time, if BLLI_id3 SVE or BHLI SVE is tagged as present,
** then the subsequent bind is required.
** In the other case, the subsequent bind shouldn't be used.
*/
# define ADL_ATM_ADDR_TAG_BIT              6
# define ADL_ATM_SELECTOR_TAG_BIT          4
# define ADL_BLLI_ID3_TAG_BIT              2
# define ADL_BHLI_ID_TAG_BIT               0


typedef enum
{
    ADL_ABSENT                             = 0x00,
    ADL_ANY                                = 0x01,
    ADL_PRESENT                            = 0x02
} ADL_tag_t;


# define ADL_TAG_MASK                      0x03


# define ADL_SVE_TAGS_OFF         0x00


# define ADL_SVE_TAGS_SIZE         1
```

```
typedef uint8_t    ADL_SVE_tags_t[ADL_SVE_TAGS_SIZE],

     *ADL_SVE_tags_ptr_t;


/*

** BLLI layer 2 protocol Tag and Value

**

** (Option 1 and 2)

**

**     7       6       5       4       3       2       1       0

** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

** |   1   |   1   |   0   | layer2_protocol                       |

** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

** |          User_specified_layer_2_protocol_information          |

** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

**

** (Absent or Any)

**

**     7       6       5       4       3       2       1       0

** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

** | BLLI_id2_tag  |             reserved                          |

** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

** |                       reserved                                |

** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

**

*/


/*

** General purpose

*/

# define ADL_BLLI_ID2_PROT_OFF     0

# define ADL_BLLI_ID2_USER_OFF     1
```

```
# define ADL_BLLI_ID2_TAG_BIT      6


# define ADL_BLLI_ID2_VALTAG_SIZE     2


typedef uint8_t    ADL_BLLI_id2_valtag_t[ADL_BLLI_ID2_VALTAG_SIZE],

     *ADL_BLLI_id2_valtag_ptr_t;


/*

** Option 1 layer2_protocol values

** User_specified_layer_2_protocol_information is padding

*/


# define ADL_BASIC_MODE_ISO_1745                    0xc1

# define ADL_ITU_Q_921                              0xc2

# define ADL_ITU_X_25_LINK_LAYER                    0xc6

# define ADL_ITU_X_25_MULTILINK                     0xc7

# define ADL_EXTENDED_LAPB_FOR_HALF_DUPLEX_OP       0xc8

# define ADL_HDLC_ARM_ISO_4335                      0xc9

# define ADL_LAN_LOGICAL_LINK_CONTROL_ISO_8802_2    0xcc

# define ADL_ITU_X_75_SINGLE_LINK_PROCEDURE         0xcd

# define ADL_ITU__Q_922                             0xce

# define ADL_ISO_7776_DTE_DTE_OPERATION             0xd1


/*

** Option 2 layer2_protocol values

** User_specified_layer_2_protocol_information is meaningfull

*/

# define ADL_ID2_USER_SPECIFIED                     0xd0


/*

** BLLI layer 2 protocol Tag and Value

**
```

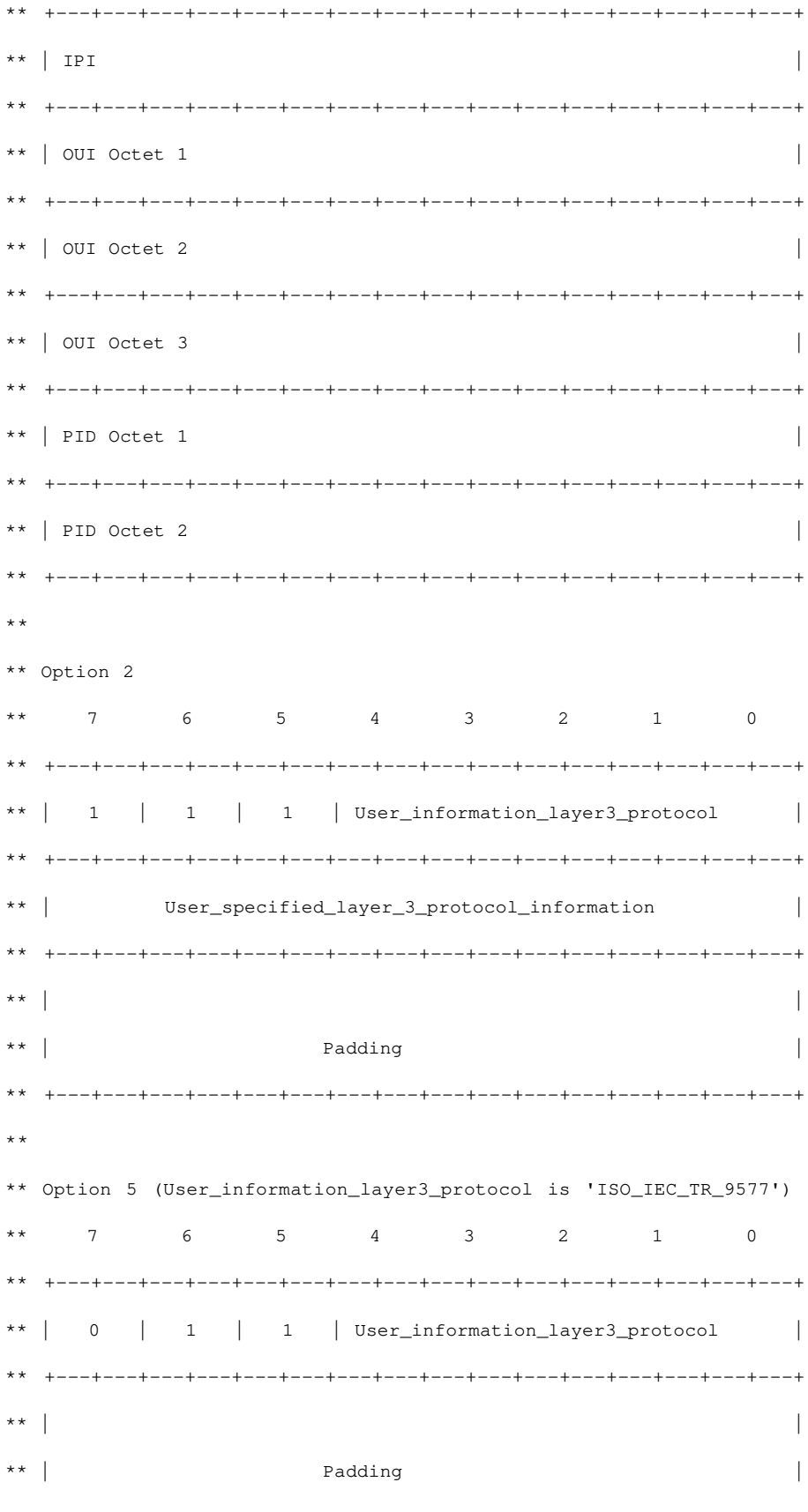
```
** (When used for PVCs)
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |   1   |   0   |   0   |   0   |   0   |   0   |   0   |   0   |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |         DLS_User_Code                                         |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
*/


# define ADL_BLLI_ID2_ID_OFF       0
# define ADL_DLS_USER_CODE_OFF     1


/*
** BLLI layer 2 protocol Tag encoding when used for PVC
*/
# define ADL_PVC                   0x80


# define ADL_SIGNALING_PVC    0xff
# define ADL_ILMI_PVC         0xfe
# define ADL_IP_PVC           0xfd                /* RFC 1577 */
# define ADL_RESERVED_HI      0xfc
# define ADL_RESERVED_LO      0x01
# define ADL_DA_PVC           0x00                /* Direct Access */


/*
** BLLI layer 3 protocol Value
**
** Options 1, 3, 4
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |   1   |   1   |   1   | User_information_layer3_protocol      |
```


** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | IPI |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | OUI Octet 1 |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | OUI Octet 2 |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | OUI Octet 3 |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | PID Octet 1 |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | PID Octet 2 |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Option 2
** 7 6 5 4 3 2 1 0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | 1 | 1 | 1 | User_information_layer3_protocol |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | User_specified_layer_3_protocol_information |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |
** | Padding |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
**
** Option 5 (User_information_layer3_protocol is 'ISO_IEC_TR_9577')
** 7 6 5 4 3 2 1 0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** | 0 | 1 | 1 | User_information_layer3_protocol |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |
** | Padding |

```
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

**

*/


/*

** General purpose

*/

# define ADL_BLLI_ID3_PROT_OFF     0

# define ADL_BLLI_ID3_IPI_OFF      1

# define ADL_BLLI_ID3_USER_OFF     1

# define ADL_BLLI_ID3_OUI_OFF      2

# define ADL_BLLI_ID3_OUI_SIZE     3

# define ADL_BLLI_ID3_PID_OFF      5

# define ADL_BLLI_ID3_PID_SIZE     2


# define ADL_BLLI_ID3_VAL_SIZE     7


typedef uint8_t    ADL_BLLI_id3_val_t[ADL_BLLI_ID3_VAL_SIZE],

     *ADL_BLLI_id3_val_ptr_t;


/*

** Option 1 User_information_layer3_protocol values

** no IPI, OUI, PID

*/

# define ADL_ITU_RECOMMENDATION_PACKET_LAYER     0xe6

# define ADL_ISO_IEC_8208                        0xe7

# define ADL_X_223_ISO_8878                      0xe8

# define ADL_ISO_IEC_8473                        0xe9

# define ADL_ITU_T_70_MINIMUM_NETWORK_LAYER      0xea


/*

** Option 2 User_information_layer3_protocol values
```

```
*/
# define ADL_ID3_USER_SPECIFIED                    0xf0

/*
** Option 3, 4 User_information_layer3_protocol values
** Option 3, IPI is not IEEE 802.1 SNAP, no OUI and PID
** Option 4 IPI is IEEE 802.1 SNAP, OUI and PID meaningfull
*/
# define ADL_ISO_IEC_TR_9577                       0xeb

# define ADL_IEEE_802_1_SNAP                       0x80

/*
** Option 5 User_information_layer3_protocol values
*/
# define ADL_ISO_IEC_TR_9577_NO_IPI                0x6b

/*
** BHLI Value
**
** Options 1,2,3
**
**     7       6       5       4       3       2       1       0
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |   1   | High_Layer_Information_Type                           |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
** |  Option 1: 8 Bytes                                            |
** |  Option 2: 8 Bytes                                            |
** |  Option 3: 7 Bytes + 1 Byte padding                           |
** +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
*/
```

```
# define ADL_BHLI_ID_TYPE_OFF      0

# define ADL_BHLI_ID_INFO_OFF      1

# define ADL_BHLI_ID_INFO_SIZE     8


# define ADL_BHLI_ID_VAL_SIZE      9


typedef uint8_t    ADL_BHLI_id_val_t[ADL_BHLI_ID_VAL_SIZE],
     *ADL_BHLI_id_val_ptr_t;


/*
** Option 1, 2, 3 High_Layer_Information_Type values
*/
# define ADL_ISO                   0x80    /* Option 1 */
# define ADL_USER_SPECIFIC         0x81    /* Option 2 */
# define ADL_VENDOR_SPECIFIC       0x83    /* Option 3 */


/*
** Combined Tags and Values in:
** - DLSAP
** - SAP (BIND)
** - Subsequent SAP (SUBS_BIND, Hierarchical)
** - Full SAP (SAP + Subsequent SAP)
** - Physical Address
*/


/*
** - DLSAP
** SVE tags and values offsets
** DLPI address components offsets
** ATM NSAP or E.164 offsets
*/
```

```
# define ADL_DLSAP_ATM_ADDR_OFF    0

# define ADL_DLSAP_ATM_SELECTOR_OFF  ( ADL_DLSAP_ATM_ADDR_OFF + \

     ADL_ATM_ADDR_VAL_SIZE)

# define ADL_DLSAP_SVE_TAGS_OFF  ( ADL_DLSAP_ATM_SELECTOR_OFF + \

     ADL_ATM_SELECTOR_VAL_SIZE)

# define ADL_DLSAP_BLLI_ID2_OFF  ( ADL_DLSAP_SVE_TAGS_OFF + \

     ADL_SVE_TAGS_SIZE)

# define ADL_DLSAP_BLLI_ID3_OFF  ( ADL_DLSAP_BLLI_ID2_OFF + \

     ADL_BLLI_ID2_VALTAG_SIZE)

# define ADL_DLSAP_BHLI_ID_OFF  (   ADL_DLSAP_BLLI_ID3_OFF + \

     ADL_BLLI_ID3_VAL_SIZE)


# define ADL_PHYS_OFF             ADL_DLSAP_ATM_ADDR_OFF

# define ADL_SAP_OFF              ADL_DLSAP_ATM_SELECTOR_OFF

# define ADL_SUB_SAP_OFF          ADL_DLSAP_BLLI_ID3_OFF

# define ADL_FULL_SAP_OFF         ADL_SAP_OFF


# define ADL_ATM_ADDR_OFF  ( ADL_DLSAP_ATM_ADDR_OFF + \

     ADL_ATM_ADDR_PREFIX_OFF)


# define ADL_DLSAP_SIZE  (   ADL_ATM_ADDR_VAL_SIZE + \

     ADL_ATM_SELECTOR_VAL_SIZE + \

     ADL_SVE_TAGS_SIZE + \

     ADL_BLLI_ID2_VALTAG_SIZE + \

     ADL_BLLI_ID3_VAL_SIZE + \

     ADL_BHLI_ID_VAL_SIZE)


typedef uint8_t    ADL_dlsap_t[ADL_DLSAP_SIZE],

     *ADL_dlsap_ptr_t;


/*

** - SAP
```

```
*/
# define ADL_SAP_ATM_SELECTOR_OFF 0
# define ADL_SAP_SVE_TAGS_OFF (   ADL_SAP_ATM_SELECTOR_OFF + \
     ADL_ATM_SELECTOR_VAL_SIZE)
# define ADL_SAP_BLLI_ID2_OFF (   ADL_SAP_SVE_TAGS_OFF + \
     ADL_SVE_TAGS_SIZE)

# define ADL_SAP_SIZE (   ADL_ATM_SELECTOR_VAL_SIZE + \
     ADL_SVE_TAGS_SIZE + \
     ADL_BLLI_ID2_VALTAG_SIZE)

typedef uint8_t    ADL_sap_t[ADL_SAP_SIZE],
     *ADL_sap_ptr_t;

/*
** - Subsequent SAP (SUBS_BIND, Hierarchical)
*/
# define ADL_SUB_SAP_BLLI_ID3_OFF 0
# define ADL_SUB_SAP_BHLI_ID_OFF ( ADL_SUB_SAP_BLLI_ID3_OFF + \
     ADL_BLLI_ID3_VAL_SIZE)

# define ADL_SUB_SAP_SIZE ( ADL_BLLI_ID3_VAL_SIZE + \
     ADL_BHLI_ID_VAL_SIZE)

typedef uint8_t    ADL_sub_sap_t[ADL_SUB_SAP_SIZE],
     *ADL_sub_sap_ptr_t;

/*
** - Physical Address
*/
# define ADL_PHYS_SIZE  ADL_ATM_ADDR_VAL_SIZE
```

```
typedef uint8_t        ADL_phys_t[ADL_PHYS_SIZE],
    *ADL_phys_ptr_t;

/*
**-------------------------------------------------------
**-------------------------------------------------------
** Quality of Data Link Service
**
** The combination of some fields are not allowed.
** The UNI 3.0 (3.1) specifies which fields combinations
** are allowed.
**-------------------------------------------------------
**-------------------------------------------------------
*/

/*
**-------------------------------------------------------
** The Tag is used because some parameters are optional or
** the DLS User wants to use the default values.
**
** In a request or response the three values are allowed.
** In an indication or confirmation, default is not allowed.
**-------------------------------------------------------
*/
typedef enum
{
    ADL_VALID,
    ADL_INVALID,
    ADL_DEFAULT
} ADL_qos_tag_t;

typedef struct
```

```
{
    ADL_qos_tag_t          tag;
    uint8_t                value;
} ADL_tagged_8bits_t;

typedef struct
{
    ADL_qos_tag_t          tag;
    uint16_t               value;
} ADL_tagged_16bits_t;

typedef struct
{
    ADL_qos_tag_t          tag;
    uint32_t               value;
} ADL_tagged_32bits_t;

/*
**-------------------------------------------------------
** ATM Adaptation Layer Parameters
**-------------------------------------------------------
*/

/*
** AAL type encoding
*/
# define ADL_AAL_TYPE_1_FOR_VOICE 0x00
# define ADL_AAL_TYPE_1            0x01
# define ADL_AAL_TYPE_5            0x05
# define ADL_USER_DEFINED_AAL      0x10

/*
```

```
**-------------------------------------------------------

** ATM Adaptation Layer Parameters

** AAL type 1 for voice

**-------------------------------------------------------

*/


typedef struct

{

    ADL_qos_tag_t          tag;

    uint8_t                AAL_Type;    /* AAL_type_1_For_Voice */

} ADL_VoiceAAL1_Parameters_t;


/*

**-------------------------------------------------------

** ATM Adaptation Layer Parameters

** AAL type 1

**-------------------------------------------------------

*/


/*

** AAL1 subtype encoding

*/

# define ADL_AAL1_NULL                    0x00

# define ADL_AAL1_VOICE_BAND              0x01

# define ADL_AAL1_CIRCUIT_TRANSPORT       0x02

# define ADL_AAL1_HIGH_QUAL_AUDIO         0x04

# define ADL_AAL1_VIDEO                   0x05


/*

** AAL1 cbr rate

*/

# define ADL_AAL1_CBR_64                  0x01
```

```
# define ADL_AAL1_CBR_1544                0x04

# define ADL_AAL1_CBR_6312                0x05

# define ADL_AAL1_CBR_32064               0x06

# define ADL_AAL1_CBR_44736               0x07

# define ADL_AAL1_CBR_97728               0x08


# define ADL_AAL1_CBR_2048                0x10

# define ADL_AAL1_CBR_8448                0x11

# define ADL_AAL1_CBR_34368               0x12

# define ADL_AAL1_CBR_139264              0x13


# define ADL_AAL1_CBR_NX64                0x40


/*

** Clock Recovery Type encoding

*/

# define ADL_AAL1_TIMING_RECOVERY_NULL         0x00

# define ADL_AAL1_TIMING_RECOVERY_SRTS         0x01

# define ADL_AAL1_TIMING_RECOVERY_ADAPTIVE     0x02


/*

** Error Correction Type encoding

*/

# define ADL_AAL1_NULL_EC                  0x00

# define ADL_AAL1_INTERLEAVING_FEC_EC      0x01

# define ADL_AAL1_FORWARD_EC               0x02


typedef struct

{

    ADL_qos_tag_t          tag;

    uint8_t                AAL_Type;           /* AAL_type_1 */

    uint8_t                AAL1_subtype;
```

```
    uint8_t               AAL1_CBR_rate;
    uint16_t              AAL1_multiplier;   /* if AAL1_CBR_rate is nx64 */
    ADL_tagged_8bits_t    AAL1_clock_recovery_type;
    ADL_tagged_8bits_t    AAL1_error_correction;
    ADL_tagged_16bits_t   AAL1_structured_data_transfer;
    ADL_tagged_8bits_t    AAL1_partially_filled_cells;
} ADL_AAL1_Parameters_t;


/*
**-------------------------------------------------------
** ATM Adaptation Layer Parameters
** AAL type 5
**-------------------------------------------------------
*/


/*
** SSCS Type encoding
*/
# define ADL_NULL_SSCS                    0x00
# define ADL_SSCOP_ASSURED                0x01
# define ADL_SSCOP_NON_ASSURED            0x02
# define ADL_FR_SSCS                      0x04


typedef struct
{
    ADL_qos_tag_t         tag;
    uint8_t               AAL_Type;              /* AAL_type_5 */
    ADL_tagged_16bits_t   AAL5_fwd_max_sdu;
    ADL_tagged_16bits_t   AAL5_bak_max_sdu;
    ADL_tagged_8bits_t    AAL5_sscs_type;
} ADL_AAL5_Parameters_t;
```

```
/*
**-------------------------------------------------------
** ATM Adaptation Layer Parameters
** User Defined AAL
**-------------------------------------------------------
*/
# define ADL_MAX_AAL_INFO_SIZE     4


typedef struct
{
     ADL_qos_tag_t    tag;
     uint8_t          AAL_Type;        /* User_Defined_AAL */
     uint8_t          user_defined_aal_info_size;
     uint8_t          user_defined_aal_info[ADL_MAX_AAL_INFO_SIZE];
} ADL_UserAAL_Parameters_t;


/*
**-------------------------------------------------------
** ATM Adaptation Layer Parameters
**-------------------------------------------------------
*/
typedef struct
{
    ADL_qos_tag_t          tag;
    uint8_t                AAL_Type;
} ADL_AAL_Parameters_Head_t;


typedef union
{
     ADL_AAL_Parameters_Head_t      AAL_Parameters_Head;
     ADL_VoiceAAL1_Parameters_t     VoiceAAL1_Parameters;
     ADL_AAL1_Parameters_t          AAL1_Parameters;
```

```
    ADL_AAL5_Parameters_t          AAL5_Parameters;
    ADL_UserAAL_Parameters_t       UserAAL_Parameters;
} ADL_AAL_Parameters_t;

/*
**-------------------------------------------------------
** ATM Traffic Descriptor
**-------------------------------------------------------
*/
typedef struct
{
    ADL_qos_tag_t          tag;
    ADL_tagged_32bits_t    fwd_pcr_clp0;
    ADL_tagged_32bits_t    fwd_pcr_clp1;
    ADL_tagged_32bits_t    bak_pcr_clp0;
    ADL_tagged_32bits_t    bak_pcr_clp1;
    ADL_tagged_32bits_t    fwd_scr_clp0;
    ADL_tagged_32bits_t    fwd_scr_clp1;
    ADL_tagged_32bits_t    bak_scr_clp0;
    ADL_tagged_32bits_t    bak_scr_clp1;
    ADL_tagged_32bits_t    fwd_mbs_clp0;
    ADL_tagged_32bits_t    fwd_mbs_clp1;
    ADL_tagged_32bits_t    bak_mbs_clp0;
    ADL_tagged_32bits_t    bak_mbs_clp1;
    ADL_qos_tag_t          best_effort;
    ADL_qos_tag_t          fwd_tagging;
    ADL_qos_tag_t          bak_tagging;
} ADL_ATM_Traffic_Descriptor_t;

/*
**-------------------------------------------------------
** Broadband Bearer Capability
```

```
**-------------------------------------------------------

*/


/*
** Bearer Class encoding
*/
# define ADL_BCOB_A                0x01
# define ADL_BCOB_C                0x03
# define ADL_BCOB_X                0x10


/*
** Traffic Type encoding
*/
# define ADL_BR_NO_INDICATION      0x00
# define ADL_CBR                   0x01
# define ADL_VBR                   0x02


/*
** Timing Requirement encoding
*/
# define ADL_TIMING_NO_INDICATION 0x00
# define ADL_END_TO_END_TIMING     0x01
# define ADL_NO_END_TO_END_TIMING 0x02


/*
** Susceptibility to clipping encoding
*/
# define ADL_NOT_SUSCEPTIBLE_TO_CLIPPING           0x00
# define ADL_SUSCEPTIBLE_TO_CLIPPING               0x01


/*
** User Plane connection configuration
```

```
*/
# define ADL_POINT_TO_POINT                        0x00
# define ADL_POINT_TO_MULTIPOINT                   0x01


typedef struct
{
    ADL_qos_tag_t          tag;
    uint8_t                bearer_class;
    uint8_t                traffic_type;
    uint8_t                time_req;
    uint8_t                clipping_ind;
    uint8_t                connect_config;
} ADL_Broadband_Bearer_Capability_t;


/*
**-------------------------------------------------------------
** Broadband Low Layer Information Parameters
** Note most of the BLLI is specified using the DL SAP. Only the parameters
** which have no routing purpose are specified in this structure. Those
** parameters must be consistant with the DL SAP.
**
** The BLLI Selector as specified in the
** Native ATM Services: Semantic Description Version 1.0
** is not implemented since the selection of the BLLI uses an other
** mechanism.
**-------------------------------------------------------------
*/


/*
** layer 2 Mode of operation
*/
# define ADL_NORMAL_MODE           0x01
```

```
# define ADL_EXTENDED_MODE         0x02


/*
** layer 3 mode of operation
*/
# define ADL_NORMAL_PKT_NUMBERING          0x01
# define ADL_EXTENDED_PKT_NUMBERING        0x10


typedef struct
{
    ADL_qos_tag_t                 tag;
    ADL_BLLI_id2_valtag_t          BLLI_id2_valtag;
    ADL_tagged_8bits_t             layer_2_mode;
    ADL_tagged_8bits_t             layer_2_window_size;
    ADL_qos_tag_t                  BLLI_id3_tag;
    ADL_BLLI_id3_val_t             BLLI_id3_val;
    ADL_tagged_8bits_t             layer_3_mode;
    ADL_tagged_8bits_t             layer_3_packet_size;
    ADL_tagged_8bits_t             layer_3_window_size;
} ADL_BLLI_Parameters_t;


/*
**-------------------------------------------------------------
** Called and Calling Party Subaddress
**-------------------------------------------------------------
*/
# define ADL_SUBADDR_NSAP                  0x80
# define ADL_SUBADDR_ATM_ENDADDR           0x90


typedef struct
{
    ADL_qos_tag_t                 tag;
```

```
    uint8_t                         subaddr_type;
    ADL_atm_addr_t                  subaddr;
} ADL_Subaddress_t;

/*
**-------------------------------------------------------
** Calling Party Number
**-------------------------------------------------------
*/
typedef enum
{
    ADL_PRESENTATION_ALLOWED               = 0x00,
    ADL_PRESENTATION_RESTRICTED            = 0x01,
    ADL_NUMBER_NOT_AVAILABLE               = 0x02
} ADL_presentation_ind_t;

typedef enum
{
    ADL_USER_PROVIDED_NOT_SCREENED         = 0x00,
    ADL_USER_PROVIDED_PASSED               = 0x01,
    ADL_USER_PROVIDED_FAILED               = 0x02,
    ADL_NETWORK_PROVIDED                   = 0x03
} ADL_screening_ind_t;

typedef struct
{
    ADL_qos_tag_t                  tag;
    ADL_presentation_ind_t         presentation_ind;
    ADL_screening_ind_t            screening_ind;
} ADL_Calling_Party_Number_t;

/*
```

```
**-------------------------------------------------------

** Quality of Service Parameter

**-------------------------------------------------------

*/

# define ADL_QOS_CLASS0_UNSPECIFIED        0x00

# define ADL_QOS_CLASS1                    0x01

# define ADL_QOS_CLASS2                    0x02

# define ADL_QOS_CLASS3                    0x03

# define ADL_QOS_CLASS4                    0x04


typedef struct

{

    ADL_qos_tag_t          tag;

    uint8_t                fwd_qos_class;

    uint8_t                bak_qos_class;

} ADL_QOS_t;


/*

**-------------------------------------------------------

** Transit Network Selection

** Each octet is an IA5 character

**-------------------------------------------------------

*/

# define ADL_MAX_NETWORK_ID_CHARS 20       /* ??? */

typedef struct

{

    ADL_qos_tag_t tag;

    uint16_t       nb_network_id_chars;

    uint8_t        network_id[ ADL_MAX_NETWORK_ID_CHARS];

} ADL_Transit_Network_Selection_t;


/*
```

```
**-------------------------------------------------------
** dl qos to use with dl_connect_request, dl_connect_indication,
** dl_connect_response, dl_connect_confirm, dl_info_ack
**
** Note only the AAL_Parameters and the first BLLI_Parameters entry
** are meaningfull in a dl_connect_response or dl_connect_confirm.
** Note only the first BLLI_Parameters entry is meaningfull in
** a dl_info_ack
**-------------------------------------------------------
*/

# define ADL_QOS_CO_ATMF_V1        3110

typedef struct
{
    uint32_t                              dl_qos_type;    /* qos_co_atmf_v1 */
    ADL_AAL_Parameters_t                  AAL_Parameters;
    ADL_ATM_Traffic_Descriptor_t          ATM_Traffic_Descriptor;
    ADL_Broadband_Bearer_Capability_t     Broadband_Bearer_Capability;
    uint32_t                              BLLI_Selector;
    ADL_BLLI_Parameters_t                 BLLI_Parameters[3];
    ADL_Subaddress_t                      Called_Party_Subaddress;
    ADL_Calling_Party_Number_t            Calling_Party_Number;
    ADL_Subaddress_t                      Calling_Party_Subaddress;
    ADL_QOS_t                             QOS;
    ADL_Transit_Network_Selection_t       Transit_Network_Selection;
} ADL_dl_qos_t, *ADL_dl_qos_ptr_t;
```