ATNP WG3/SG3 (Upper Layer Architecture)

Draft Upper Layer Guidance Material for CNS/ATM-1 Package

(Presented by WG3/SG3)

REVISION STATUS

Version 1.0 First draft Guidance Material for Upper Layers (Gold Coast)
Version 2.0 Second draft Guidance Material for Upper Layers (Bruxelles)

TABLE OF CONTENTS

1. Introduction

1.1 Scope of document

This document is intended to describe the architectural framework for the standardisation of ATN Upper Layers. It covers the following areas:

- Supporting upper layer stacks
- Upper layer overhead comparisons
- Encoding rules and data compression, for upper layer PCI
- Naming, addressing and registration
- Application layer structure and service elements
- The use of transport services

The following sections draw together existing material on each of these areas, as well as providing additional summaries and analysis not found elsewhere.

1.2 Objectives

The aim of this document is to define the general communications architecture for ATN upper layer(s) (i.e. everything above the ATN Transport Service) and to provide reference material to aid the development and implementation of the upper layers.

The basic aim of this document is to define a set of architectural principles which will allow ATN Applications to be constructed in a standard way. This "building block" approach has many well-known advantages, including:

- the duplication of effort associated with designing and debugging similar functionality for many different application types is minimised
- the same type of design problem would otherwise have to be repeatedly solved for each new application
- the productivity of designers, programmers, system engineers and testers is increased, as they only have to deal with a single architecture
- the certification effort is eased, as experience is gained with previously accredited modules.

This work should stabilise and document the architectural basis for standardisation in ICAO, in particular considering:

- The approach to upper layer stack selection
- Preferred data encoding schemes
- Naming and Addressing

- Application Layer Structure

1.2.1 The ATN Upper Layer Environment

The OSI/ATN environment as it relates to upper layers and ATN Applications is depicted in Figure 1.1. This architecture effectively embraces two areas of standardisation:

a) Communications profiles of the upper layers to deliver the communications requirements for proposed applications; such profiles should be based on existing or new (defined by the ATN community) standards.

b) Application specifications, defining the messages and message sequence rules for applications to meet specific operational requirements, as set down in a completed application SARPs.

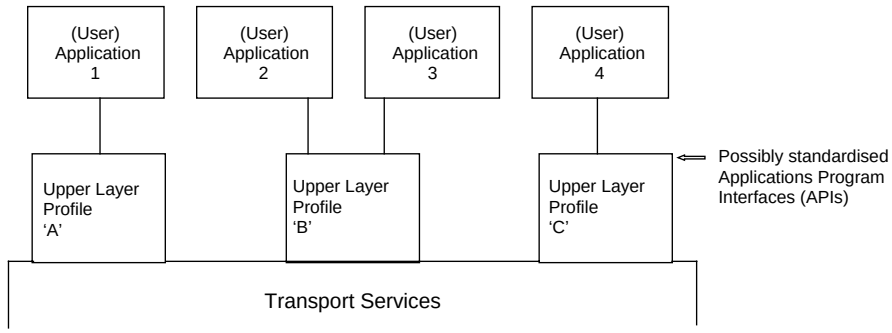


Figure 1.1 : ATN Host Software Environment

In OSI terms, the DLA as defined above will comprise the LOCAL part of an OSI Application Process (AP). The OSI AP will also include an Application Entity, comprising the Application Service Elements and Application Service Objects (ASEs and ASOs - see 2.3.4) appropriate to the specified profile.

Each DLA Application Process shall be allocated a name by which the particular application, version and message structure can be recognised, and which can be used to indicate the application support capability of a particular aircraft or ground station. The name may be used by the Context Management Application (CMA).

2. Architecture

2.1 Scope

This section examines the basic ATN Application architecture to provide a framework in which the upper layer profiles are defined. The services provided by upper layers are described for reference.

2.2 The Basic Architectural Concept

2.2.1 Scope

This section proposes generic communication services for use on the ATN. It describes the proposed services and suggests existing application layer standards which may be used to implement them. It also examines the application of the generic communication services to the ATN and considers some of the special constraints imposed on the ATN by the communications subnetworks.

2.2.2 Background

The framework for the standardisation of the upper layers puts forward the proposal that a set of standardised common communication services should be provided on which user applications, or so called Data Link Applications, would be constructed. These communication services will be used by the user applications to exchange information and, where thought appropriate, the interactions (i.e. message exchanges) which take place over these services would also be standardised in terms of the functions offered by the specific service.

The adoption of this framework means that the standardisation process may be subdivided into two parts; firstly, the standardisation of a common set of communication services, and secondly the standardisation of the DLA which uses those services. The aims of this framework are:

a) to keep to a minimum the number of standard application services

b) to use existing OSI application layer standards wherever possible, thus removing the need to define, standardise and conformance test new application layer standards

c) to tailor some of the service profiles to the underlying restrictions of some of the low bandwidth air-ground subnetworks in the ATN, but to use recognised application profiles wherever possible.

It is planned to define a limited set of communication service profiles for use in the aeronautical domain, to provide applications with access to the ATN. Each profile constitutes an upper layer "protocol stack" definition which when implemented provides the appropriate functionality in the selected upper layers.

Profiles are defined by selecting valid combinations of protocol standards and forming valid subsets in such a way as to deliver a specific level of service to the applications. A number of such profiles will be defined to provide wide applicability for the differing upper layer support requirements of different applications. Collectively, these profiles will provide a well-defined set of services which can be utilised when designing and implementing particular ATN applications. This does not imply that it is necessary or even desirable to implement the complete set of selected upper layer profiles on all end-systems. Subsets of the full set can be selected, to provide appropriate levels of functionality to meet the requirements of different classes of applications.

Inherent in this discussion is the desire to use standardised protocols whenever possible. Many of the necessary protocol standards already exist, and profiling is already being undertaken to produce International Standardized Profiles (ISPs). However, functionality or performance requirements may exist which are not satisfied by existing upper layer protocols. In such circumstances, it may be necessary to develop specialised upper layer protocol definitions, within the framework of the OSI reference model, for use in the aeronautical domain.

The adopted framework separates the communications profiling from the application standardisation and tries to standardise at the communications level only a small number of generic communications classes, which would be appropriate for use by a wide range of applications.

Each DLA will be defined by a SARPs document. The specification document will include complete message definitions, including encoding rules, sequencing rules, exception conditions and temporal relationships to be met by the application. It will also specify what Quality of Service (service characteristics) are required from the underlying communications, so as to allow selection of the appropriate ATN UL profile and Data Link.

2.3 Benefits of this approach

The following benefits result from this approach:

a) It allows the separation of application specific and communication specific functions.

b) The certification of communications and applications software may be carried out separately.

c) It allows the use of a small number of standard communication services, based on distinct modes of interaction, by a large number of DLAs.

d) It does not require the definition of new application contexts, presentation transfer syntaxes,... whenever a new DLA is introduced or an existing DLA modified.

e) Some DLA communication requirements may be satisfied by existing ASEs and profiles.

f) The communication services may in some cases be based on COTS (Commercial Off The Shelf) products.

g) It does not require the implementation of application specific interactions (eg timeouts, message sequencing rules) within the communication service.

In the ATN Protocol Architecture, OSI application entities provide communications services to ATN applications. The service boundary between the application entities and the ATN applications is an abstract interface which may or may not be realised as an exposed interface in a real implementation. ATN profiles are defined to lie on the service-provider side of this boundary (see figure 2.1).

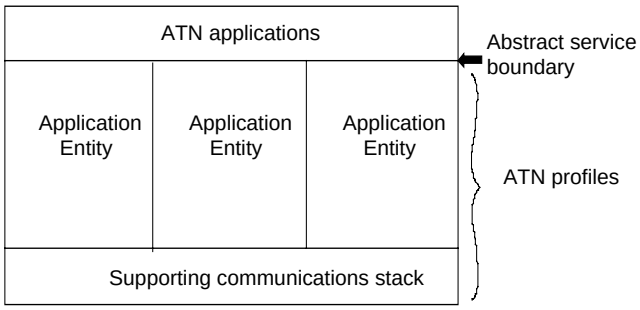


Figure 2.1 : Position of ATN Profiles within an End System

ATN applications will be defined to use the services of the selected ATN profiles. The communication aspects of these applications will be defined in terms of:

a) the semantics and structure of the information to be exchanged (‘messages’)

b) the rules governing the dialogue between parties (message sequencing)

c) requirements imposed on the underlying communications services (quality of service, etc.)

ATN Applications will be standardised as ICAO SARPs which will specify which subset of the ATN profiles is appropriate for the particular application, and if more than one subset is possible, under which circumstances one or the other would be selected. These choices map onto upper layer stack selections for the ATN application.

2.4 Upper Layer Concepts and Structure

2.4.1 Scope

This section deals with the upper layer model. It covers the existing ASE structure and also considers the applicability of the revised OSI Application Layer Structure (ALS) model in the 2nd edition of ISO/IEC 9545 (also known as Extended Application Layer Structure or XALS) in terms of ASOs and associated Control Functions.

2.4.2 Functions of the Upper Layers

The service currently offered by the ATN Internet is a low-level communications service which corresponds to the OSI Transport Service defined in ISO 8072. Although it is possible to construct ATN Applications which make direct use the raw transport service, such applications will not benefit from the "common building block" approach.

It is therefore envisaged that a set of functions which add value to the raw ATN transport service will be standardised, and these will provide high-level services to the specific ATN Applications. Since it is a fundamental principle of the ATN that it adopts the protocols defined in the international standards for Open Systems Interconnection (OSI), it is logical to look to the standards defining the upper layers of the OSI model to provide the required value-added functions.

Before examining further alternatives, this section considers what the standard OSI upper layer (Session, Presentation and Application) protocols offer as added value on top of the raw transport service. It is these features that need to be incorporated in any ATN Application, or rejected as being unnecessary for a particular requirement.

One important, static function is to define formats and encodings for data interchange, in an unambiguous and open way, i.e. independent of any particular hardware bit-ordering or word-size conventions.

The features listed in the following table are taken from a paper by Peter Furniss. They deliberately use inexact language and (sometimes) avoid technical terms as it is the effective role, not the specified function that is important:

	Upper Layer Feature	Comment
a)	The first octet of each session protocol data unit (PDU) identifies what part of the conversation is being sent	Something will <u>always</u> be needed to distinguish broad categories of message that all travel on the same supporting service (in this case, on T-DATA)
b)	Session and presentation connects include selectors that can be used for upper-layer routing.	The upper-layer selectors are redundant if the upper-layer implementation is integrated
c)	Presentation layer is primarily concerned with presentation context - the identification and negotiation of the abstract and transfer syntax of the application data. There are several aspects to this:	
	i) the presentation context identification of each piece of application data allows different message types (APDUs) to be distinguished, without fear of ambiguity, even if two messages from different sets (abstract syntaxes) happen to have the same bit-pattern when encoded.	This is only necessary when there are multiple abstract syntaxes (but, since ACSE is considered to be in the application layer, there always are).
	ii) the presentation context identification can (usually) be used to distinguish separate streams of dialogue - because each stream will usually be using a separate set of message types	This works provided all the ASEs involved are different, but becomes rather complicated otherwise. Something that is not concerned with message type identification is needed for the general case of identifying the appropriate recipient of a message. Assuming that the message type tells you who it is for does not always work (try sending an anonymous birthday card to twins!)
	iii) abstract syntax negotiation states/agrees which sets of message types will be used on the association	See also f): the application and presentation contexts have some overlap of function. Since an application context definition identifies which ASEs, and thus which sets of message types, will be involved, abstract syntax negotiation can be no more than a parameterisation of the application context, and will usually be unnecessary.
	iv) transfer syntax negotiation (for each particular presentation context) allows alternative representations of the same message to be offered	This is probably the most useful feature of the upper-layers, and the least used at present. It offers considerable scope for extensible, conformant optimisation in the processing OR the transmission of application protocols.

d)	Presentation provides bracketing around the encodings of application data, so they need not contain their own length fields	Anything whose encoding is not self-delimiting will require this feature. Conversely, if the supporting layers provide the feature, it need not be duplicated in the application protocol (i.e. the application protocol need not be self-delimiting)
e)	ACSE provides naming fields (AP-title, AE-qualifier and invocation identifiers) that can be location-independent	The ability to carry the names of the communicating entities explicitly, rather than inferring them from the addressing (via reverse lookup) is likely to be needed generally, but cannot always achieve what is hoped for. It cannot be relied on for effective authentication of the sender, though it can give a weak identification. Depending on how fast directory updates take place, it may allow a subsequent return call to the same entity on a different lower layer address. But it certainly does allow an un-authenticated assertion of who the other party is.
f)	ACSE carries an application context identifier that identifies what the communication will be about, and what should be assumed about the intents and capabilities of the partners.	In a particular instance, an application context will either be completely redundant or absolutely essential! Often the entity that is listening on a particular lower layer address will only support one application context. If a call is made to that address, the only advantage in communicating the application context explicitly is to identify "wrong numbers". In other cases, the entity may be capable of several different things, and the application context is needed to tell it what the call is about.
g)	ACSE identifies which of its APDUs is being used (they all have different tags)	Since each ACSE APDU has (at present) a unique mapping to Session, this is redundant. However, if different ACSE APDUs could travel on the same immediately supporting service, the distinction is needed - in fact it has taken on part of the role of the session SI field in a).
h)	With its recent extension, ACSE can carry security information	Where needed, and perhaps especially combined with the naming fields, carriage of security information will be vital

j)	All three protocols have facilities for version negotiation and, in the connect-request, extensibility rules that allow old implementations to ignore fields they do not recognise	Inventing a non-extensible protocol can be expected to be the last action of a doomed design team.

3. Application Layer

The application layer is the seventh and topmost layer in the OSI seven layer reference model. It makes use of the presentation service in order to cooperate with peer applications in other end systems, and provides services to the user (which may be a human or another application).

The application layer has a number of significant structural differences compared with the other layers. Each of the layers up to presentation provides a fixed standardized service to the layer above. The application layer however has many different functions (depending on the application), and may provide services either to the application user (which is equivalent to 'the layer above') or to other elements within the application layer itself.

Figure 3.1 illustrates at a high level the relationship between the various components of the application layer.

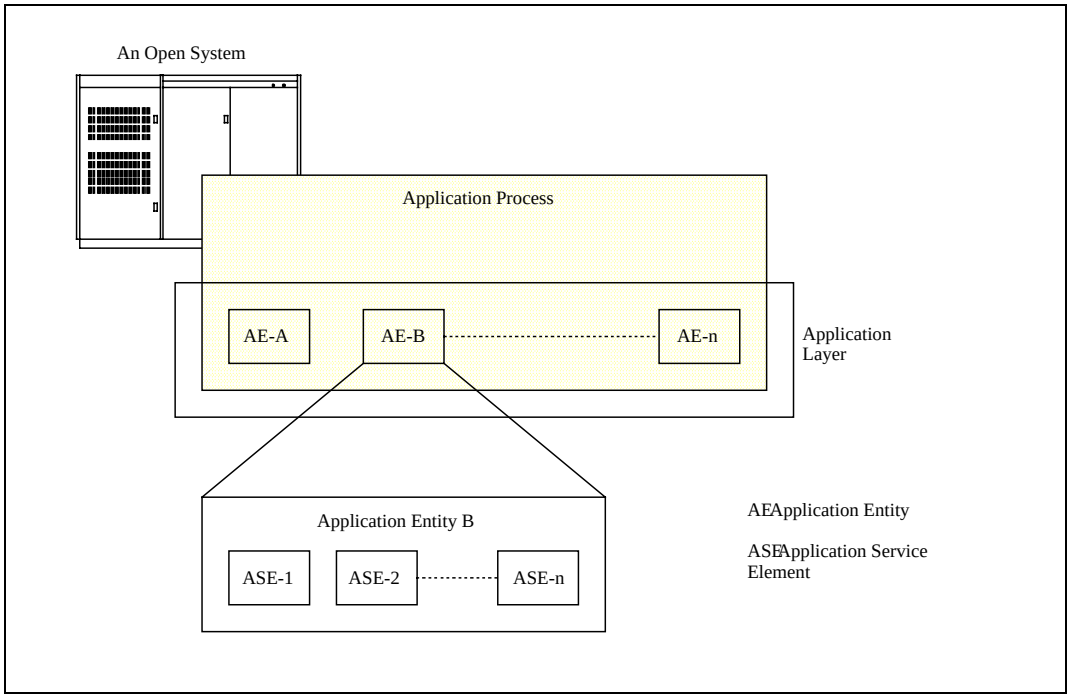


Figure 3.1 *: The Major Application Layer Components*

Figure 3.2 shows the various components of in the application layer as defined in the Application Layer Structure (ALS) model in ISO 9545, and shows how they are related.

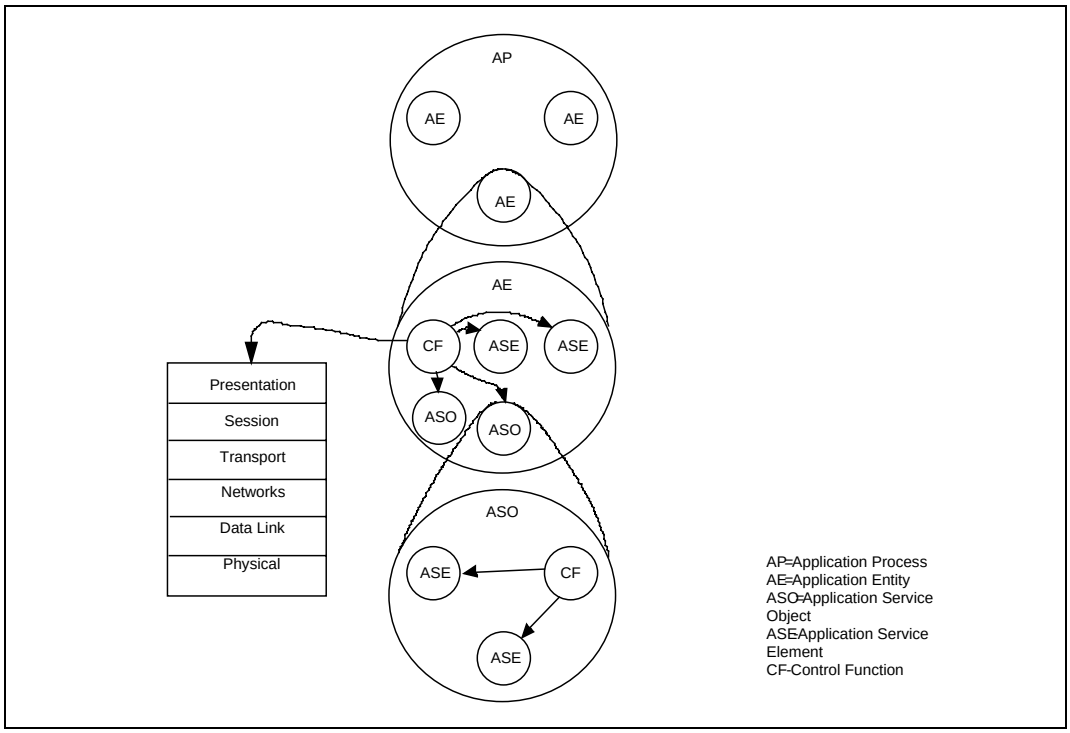


***Figure 3.2** : Components of the application layer as defined in ALS*

Associations

ASOs cannot cooperate until invoked. When two (or possibly more) ASOIs cooperate, the relationship between them is known as an ASO Association.

At any given time, an ASOI may have zero, one or more than one ASO associations with other ASOIs.

An ASO association is between two (or possibly more) ASOIs. These peer ASOIs are not necessarily of the same type, but must be of complimentary types. For example, if they are to exchange data, both must understand the same data syntax.

An ASO association is characterised by an ASO Context, which defines:

- the communications behaviour
- a set of rules and state information
- the number of ASOIs allowed in the ASO association
- how the ASO association can be started and finished

The ASOIs agree the ASO context before the ASO association is established. The ASO context may be identified by either defining it with the information listed above, or (more practically) by sending the identification of an ASO context that is well known or has been agreed beforehand.

The agreement of the ASO context is usually performed by the ACSE which is therefore one of the ASEs within the ASO.

An application association is a special type of ASO association which exists between an AE in one application and a peer AE in another. In a similar way, an application context is a special type of ASO context that characterises the application association. An application association underlies every other ASO association during the lifetime of the AEI. This is illustrated in figure 3.3.

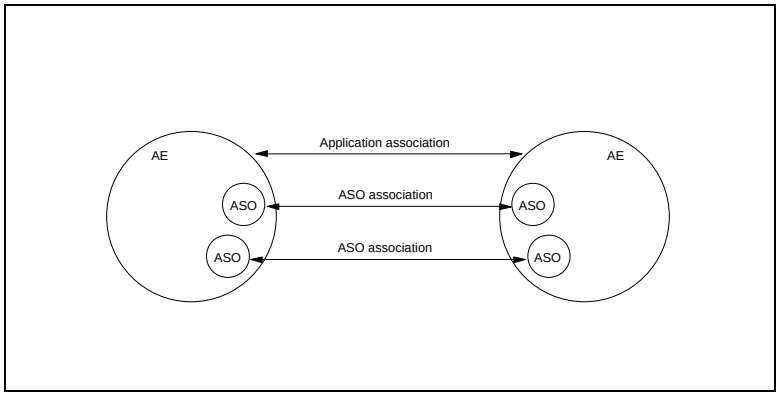


Figure 3.3 *: Application and ASO Associations*

3.1.1 Upper Layer Structure

The fact that the upper layers (Session, Presentation and Application) are divided into three layers in the OSI reference model, and that the Application Layer is divided into discrete service elements, does not imply that this abstract division should be visible in real world implementations.

To assist in application design, a standardized application layer architecture has been developed as ISO/IEC 9545 Application Layer Structure (ALS). The architecture specifies the existence of the following major application layer components:

- *Application Process (AP)*: an element within a real Open System which performs the processing for a particular application. An example of an AP is an X.400 software package, of which some elements will be responsible for OSI communication and other elements will have responsibilities beyond the scope of the OSI environment, for example user interfaces and document filing systems. The AP can thus be considered to be partially in the OSI environment and partially in the local system environment. It may be modelled as a set of application entities (AEs).

- *Application Entity (AE)*: an aspect of an application process pertinent to OSI, and is the means by which application processes exchange information using defined application protocols and the underlying presentation service. An example of an application entity is an X.400 Message Transfer Agent (MTA) or Message Store (MS).

- *Application Service Object (ASO)*: the generic term for an object which performs some communications related task. For example, a file transfer system such as an FTAM initiator is an ASO. An AE is a particular kind of ASO, the outermost ASO. An ASO can contain further ASOs, which are thus recursively defined.

- *Application Service Element (ASE)*: An AE can be broken down further into a number of ASEs, each of which provide a set of OSI communication functions for a particular purpose. An ASE is a particular kind of ASO which is elemental and therefore cannot be subdivided. An ASE may be thought of as a leaf node in a tree of ASOs. ASEs may provide general purpose functions which can be used by a number of applications. ASEs in general are defined in separate standards. For example, the *Association Control Service Element (ACSE)* is used to create and release associations between applications, the *Remote Operation Service Element (ROSE)* is used to support general-purpose enquiry-response type interactions, and the *Reliable Transfer Service Element (RTSE)* is used to ensure that a unit of information is completely transferred once and only once between communicating applications.

- *Control Function (CF)*: this exists within an ASO to coordinate the use of the different services provided in the ASOs and ASEs, and also the use of external services such as the presentation service. It provides a mapping of the ASO to the subordinate ASOs and ASEs which it contains.

AEs, ASOs and ASEs are abstract representations of some part of an application. They cannot perform any action without first being invoked; this is equivalent to having a computer program that cannot do anything until it is run. An *AE invocation (AEI)* can be thought of as an AE that is running; similarly an *ASO-invocation (ASOI)* and *ASE-invocation (ASEI)* are ASOs and ASEs that are running.

3.2 Upper Layer Services and Protocols

Application layer standards define the services and protocols supported by ASEs. Some of these standards define services which are common to a range of application layer standards, and which can be used as "building blocks" when defining new applications. The most significant of these are summarised below; others define the specific services and protocols supported by particular application layer standards such as electronic messaging and file transfer.

Association Control Service Element (ACSE)

The ACSE service allows one application-specific AE (for example, an X.400 MTA) to request that an association be established with a remote AE (in this example, an adjacent X.400 MTA) for the purpose of information transfer. ACSE will attempt to establish the association using the supporting OSI layers, and will report the success or failure of this attempt to the requesting AE. ACSE also provides for the orderly and abrupt release of the association once established.

The connection-oriented ACSE service and protocol are defined in ISO 8649 (CCITT X.217) and ISO 8650 (CCITT X.227) respectively. The connectionless ACSE protocol is defined in ISO 10035 (CCITT X.237), in which the request to establish an association, the information to be exchanged and the request to release the association are all transferred together.

The following ACSE functional units are defined:

- Kernel
- Authentication
- Application Context Negotiation

ASO-Association Control Service Element (AACSE, or "A2CSE")

There is currently a project under way within ISO/IEC JTC 1/SC 21 to modify the ACSE standards to support the revised ALS model more directly by allowing associations between named ASOs to be set up explicitly and to perform the context demarcation functions required for ASO-associations

Remote Operations Service Element (ROSE)

ROSE provides a general-purpose 'request-response' service for use by application-specific ASEs. It is therefore used by interactive-type applications in which one ASE requests that another ASE performs an operation, with the performing ASE reporting the outcome of that operation to the requesting ASE. For example, ROSE is used by the X.400 remote User Agent (UA) to interrogate and retrieve messages from the Message Store.

The ROSE service and protocol are both defined in ISO 13712 (CCITT X.218 for the service and X.228 for the protocol). Connectionless ROSE is defined in ISO/IEC 13712-2/PDAM 1.

3.3 Encoding Rules and Data Compression

3.3.1 Scope

This section covers the encoding of information for interchange between heterogeneous systems, and possible compression mechanisms. There are three areas where data compression could be carried out:

- in application specific user data
- in the presentation layer for APDU overhead reduction
- at the link layer where standard compression techniques could be used (eg V.42bis)

Abstract Syntax Notation One (ASN.1) is a standard notation for representing data structures in an implementation-independent way[1]. The standards for ASN.1 have recently been updated. There exist a number of standardised encoding rules which enable data structures expressed in ASN.1 to be encoded as a machine-independent octet stream which is well suited to data interchange. Most current OSI upper layer protocols are specified using ASN.1 definitions, and they explicitly specify the Basic Encoding Rules (BER) for encoding their protocol data units (PDUs). The use of the Packed Encoding Rules (PER) might be more efficient but would not conform to current OSI standards. This issue needs to be investigated, with the possible outcome of requesting alternative encoding support in ISO standards.

3.3.2 Standard Encoding Rules

Before an application transmits data to a peer on another machine, the data must be encoded in some form. The data will already be stored on the local system and encoded in some local format, but this is not necessarily acceptable to the peer application. It may, therefore, have to be re-encoded before being transmitted. On receiving the data, the peer application may have to reformat it into some local form suitable for local use. The data transferred between the two systems uses some common encoding for transfer, which is the subject of this section.

1 The fact that it is called ASN number 1 emphasises that other syntax notations are foreseen in the future.

Before the definition of the presentation layer (e.g. in X.400 1984 and in Internet applications) the encoding rules had to be built into the application itself. Thus there was no possibility of choosing a more efficient encoding without changing the definition of the application. One of the functions of the presentation layer is to negotiate the set of transfer encodings to be used.

The application need not be concerned with the encoding rules, it simply deals with information in its local syntax, which must be a realisation of the abstract syntax in which the application is defined. ASN.1 (Abstract Syntax Notation number one) is commonly used to define the syntax of information in an abstract form, divorced from the concrete syntax of the encoding rules. The ASN.1 standards describe the notation which is used in the specification of existing ASEs, and which can be used to develop new ASEs or to express application user data.

Several sets of standard encoding rules have been defined or proposed, including the following:

Basic Encoding Rules (BER)

BER is the original set of ASN.1 encoding rules, assumed by many existing OSI standards for their transfer syntax. The encoding is based on an octet-aligned TLV (type-length-value) approach. The Basic Encoding Rules provide a number of ways to encode ASN.1, giving the encoding entity a choice of methods of encoding lengths, and in some cases a choice of order. BER makes no attempt to change the encoding of long octet strings in order to reduce the size.

Packed Encoding Rules (PER)

PER has been designed to minimise the bits transmitted. The encoding rules work on the basis that every bit that is not needed to transmit useful information is not sent. Thus for example, a Boolean is transmitted as a single bit rather than as an octet. Tags are generally not included and other ways of encoding this information are used instead, for example in a SEQUENCE with OPTIONAL and DEFAULT elements a single bit index is usually all that is needed to indicate the presence or absence of each element. There are four variants, namely:

- Packed encoding of a single ASN.1 type (basic aligned)
- Packed encoding of a single ASN.1 type (basic unaligned)
- Packed encoding of a single ASN.1 type (relay-safe-canonical aligned)
- Packed encoding of a single ASN.1 type (relay-safe-canonical unaligned)

The ALIGNED variety of PER inserts padding bits to align some elements to octet boundaries, whereas the UNALIGNED version does not. In the basic aligned and basic unaligned variants, the number of bits is minimised, but it is assumed that both encoder and decoder have precise foreknowledge of the abstract syntax being encoded, including all constraints. Thus, the flexibility of BER to handle unknown

data types by skipping over them is lost. In some cases, the use of PER may result in a larger encoding and decoding processing overhead, compared with BER.

3.4 ASN.1 PER Backgrounder

The Packed Encoding Rules (PER) are defined in ISO/IEC 8825-2. PER makes use of features which appear in later versions of the ASN.1 syntax (constraint specification, EMBEDDED PDV, etc) to minimise the number of bits encoded for interchange. In each interchange, the receiver must have *a priori* knowledge of the types of data being sent for it to be possible to decode the data stream.

There are a number of variants of PER, namely BASIC-PER and RELAY-SAFE-CANONICAL-PER which come in both an ALIGNED and UNALIGNED variants. In the ALIGNED variant, padding bits are inserted from time to time to restore octet alignment before character strings and large integer values. In the UNALIGNED variant, which was added at a late stage in PER approval, no such padding bits are inserted.

For the purpose of this document the term ASN.1/PER is used to indicate the packed encoding of a single ASN.1 type (basic unaligned), which has been registered as:

{joint-iso-ccitt asn1 (1) packed-encoding (3) basic (0) unaligned (1)}

The specification of PER requires that some fields are *bit aligned* (bit-fields) and other fields are *octet aligned* (octet-aligned-bit-fields). Note that the ALIGNED variants of PER uses a B‘0’ as alignment (padding) bits. There follows a simplified review of the techniques involved in using Packed Encoding Rules.

The following terms are used in ASN.1/PER:

Preamble - is a bit-field with a bit corresponding to each OPTIONAL or DEFAULT item - if the entry is present the bit is set to ‘1’ otherwise it is set to ’0’.

Canonical order - indicates that the elements are sorted as follows: Universal class tags first, followed by Application-wide, Context-specific and Private-use tags then within each class elements are sorted in ascending order of their tag number. A selection of types is shown below.

Octet-aligned-bit-field - a product of encoding that is a sequence of bits that begin on an octet boundary but need not finish on one.

Bit-field - a product of encoding that is a number of bits that may or may not begin on an octet boundary.

Field-list - a set of bit-fields and octet-aligned-bit-fields. Each value is encoded to form either a octet-aligned-bit-field or a bit-field then the field is appended to the current field-list (being preceded by a length indicator is some cases).

Constrained - a number that has a known (or determinable) lower and upper bound. The use of bounded values assumes that the decoder has the same type definitions

and can deduce the same values for the bounds as the encoder. This will most likely be specified through a profile if not explicit in the original defining standard.

Unconstrained - a number that has a unknown (or undeterminable) lower and upper bound.

Semi-constrained - a number that has a known (or determinable) lower bound but an unknown (or undeterminable) upper bound. The use of the lower bound value assumes that the decoder has the same type definitions and can deduce the same value for the lower bound as the encoder. This will most likely be specified through a profile if not explicit in the original defining standard.

The encoding of a component of a data value consists of one of the following, where encoding b) applies where the contents are large:

a) three parts (any or all of which may be empty, bit-fields, or octet-aligned-bit-fields) which appear in the following order:
   i) a preamble
   ii) a length determinant
   iii) contents

b) an arbitrary number of parts as follows:
   i) a preamble
   ii) a length determinant for the first fragment of the contents (octet-aligned-bit-field)
   iii) the first fragment of the contents (octet-aligned-bit-field)
   iv)... repeat ii) and iii) for all other fragments of the contents.

Type Encoding

ASN.1 PER does not use tags per se. It forms a canonical ordering of the alternatives and then uses a preamble as an indicator. The preamble uses a bit string to represent the presence (B'1') or omission (B'0') of DEFAULT or OPTIONAL items. A recipient of a PER-encoded data stream must know *a priori* the abstract syntax of the data being encoded.

Length Determinant Encoding - The length determinant is used, when required, to indicate the number of components in a SEQUENCE OF or the number of bits or octets in a data value. When a length determinant is present it can be encoded in a number of ways, but it will always be either constrained or semi-constrained with lower bound >= 0.

· **Constrained** The length determinant is treated as an Integer whose value corresponds to the offset from the lower bound, and may therefore be a bit-field or an octet-aligned-bit-field.

· **Semi-constrained** The length determinant is treated as an Integer whose value corresponds to the offset from the lower bound. The length is always octet-aligned. If the length is less than 128 it is encoded as B'0xxxxxxx', if the length is less than 16384 (2^{+14}) it is encoded as B'10xxxxxx xxxxxxxx' otherwise it is

encoded using a length of lengths introducer octet followed by the length octets B‘11yyyyyy xxxxxxxx ... xxxxxxxx’.

Value Encoding - Each type has its own methods of encoding.

· **Boolean** A Boolean type will occupy just one bit and immediately follows the current field-list. B‘0’ indicates FALSE and B‘1’ indicates TRUE.

· **Integer** An integer type has a number of options:

Constrained - A constrained integer type is specified as INTEGER (n..m). The lower bound (n) is subtracted from the value of the integer before encoding. A value in the range 0..255 uses a bit-field, a value equal to 256 uses 1 octet, a value in the range 257..65536 uses a 2 octet octet-aligned-bit-field and a value greater than 65536 uses indefinite length encoding and an octet-aligned-bit-field.

Semi-Constrained - A semi-constrained integer type is one whose values are constrained to exceed or equal some value "lb" , with no upper bound being specified. The lower bound (lb) is subtracted from the value of the integer before encoding, and indefinite length encoding is used.

Unconstrained - Indefinite length encoding is used, and the value is encoded as a 2’s-complement binary integer in an octet-aligned bit field with the minimum number of octets.

· **Bit String** If the bit string is constrained and contains up to (and including) 16 bits it is a bit-field. Larger bit strings are octet-aligned-bit-fields. If it is constrained there is no length, otherwise include a length.

· **OctetString** If the length is zero it is not encoded. If the length is fixed at 1 or 2 then it is not an octet-aligned-bit-field and has no length field. If it is a fixed length octet string then encode with no length octets. Otherwise for unconstrained octet strings the (unconstrained) length is present.

· **Null** No encoding

· **Object Identifier** It uses the BER encoding (octet-aligned-bit-field) preceded by a semi-constrained length (with a lower bound of 1).

· **IA5String** If the length is zero it is not encoded. If the length is fixed at 1 or 2 then it is a bit-field with no length. If it is a fixed length octet string then the (constrained or semi-constrained) length is present. Otherwise for unconstrained octet strings the (unconstrained) length is present.

· **Enumerated** Sort into ascending order, then index from 0. Then treat the index as an Integer.

· **Sequence** A preamble is used as an introducer when OPTIONAL or DEFAULT components are defined in the SEQUENCE. The presence or absence of

each such component is indicated in the preamble bit-map by a ‘1’ or ‘0’ respectively. It is followed by the encodings for the individual elements.

· **Sequence Of** If the number of components is fixed there is no length field, otherwise a length field is present. It is followed by the encoding of each component in turn.

· **Set** The Set type has its elements sorted into a canonical order and is then encoded as if it had been declared a Sequence.

· **Choice** The Choices are assigned an index (the choice index) starting at 0 for the first and ‘n’ for the last. The choice index is then encoded as an Integer with a range of 0..n.

· **PDV-type** A preamble is used as an introducer, with bits 8 and 7 set to B‘1’ the lower six bits are then used to select the Presentation Context Identifier. This is followed by a length and then the actual PDV.

· **Real** The Real type is encoded as BER.

Distinguished and Canonical encoding rules (DER and CER)

For each ASN.1 type, these encoding rules select only one encoding from those allowed by the BER, eliminating all of the sender’s options. The bit efficiency is therefore comparable with BER. Two main reasons for using these in preference to BER are: (a) some data integrity algorithms require the information to be encoded in exactly the same way whatever the implementation, and (b) skinny stack implementations which include the application layer will require a standard encoding of information so that pattern matching techniques can be used to scan incoming PDUs for known bit sequences. The main difference between DER and CER is that DER uses definite length encoding and CER uses indefinite length. CER is best chosen when the information is large and not all in main memory, whereas DER is best chosen when the information is all in memory (and thus it is easy to determine the size).

Lightweight encoding rules (LWER)

This is a proposed set of rules which would minimise the overhead of encoding and parsing the resulting transfer data streams in real time. Since most applications work on the basis of transferring data that is contained in a data structure in memory, LWER is designed to transfer data structures. The ASN.1 representation of the PDU will be transformed into a data structure in a systematic way. Provided that both applications use this structure and provided they have a similar architecture (in particular the same word size), the data can be loaded into memory and extracted from memory with minimal effort. The number of bits transferred will probably be higher than for BER. The intended environment is a high bandwidth communications path.

3.5 Naming, addressing and registration

3.5.1 Scope

This section contains material on upper layer naming and addressing which is not covered in the ATN Manual (2nd edition).

3.5.2 Naming and Addressing Guidance

Naming refers to the identifier which must be assigned to any information object which may need to be referred to during information processing. Addressing refers to the physical location of a resource. To quote ISO/IEC TR 10730:

> *"Naming and addressing mechanisms are an essential aspect of OSI. Real Open Systems, even while being fully conformant to OSI protocols in all seven layers, may well be unable to set up a dialogue because of inconsistencies among their naming and addressing policies."*

3.5.3 Naming

Information objects which may need to be named include:

- application processes
- managed objects
- ATN application message types

AP, AE and Context Naming

In a given end system, application processes (APs) are the elements which perform the information processing for particular user applications. They are identified by AP-titles, which must be unambiguous throughout the OSI environment (OSIE), and thus require a Registration Authority to allocate names.

When APs in different end systems need to cooperate in order to perform information processing for a particular user application, they include and make use of communication capabilities which are conceptualised as application entities (AEs). An AP may make use of communication capabilities through one or more AEs, but an AE can belong to only one AP.

Application entities are each named in terms of a unique Application Entity Title (AET), which consists of an AP-title and an AE Qualifier.

AP-titles and AE-qualifiers may be assigned either an attribute-based name form or an Object Identifier name form. When an AP-title is allocated an attribute-based name form, all of the associated AE-qualifiers must also be assigned an attribute-based name form; when an AP-title is allocated an Object Identifier name form, all of the associated AE-qualifiers must also be assigned an Object Identifier name form.

It may at times be necessary to make a distinction between the various invocations of a given AP running concurrently on an Open System. Thus it is possible, say, to have a pool of generic application servers, and when a server is allocated to perform

a particular task, it is identified via its Invocation-identifier. This is done through the use of AP-invocation-identifiers which must be unambiguous only within the scope of the AP, and thus do not have to be registered.

Similarly, it may be necessary to distinguish between the various invocations of a given AE running concurrently as part of a given AP. This is done through the use of AE-invocation-identifiers which must be unambiguous within the scope of the {AP-invocation, AE} pair and thus do not have to be registered.

For communication purposes, AE-invocations have to handle one or more Application Associations. These can be identified by Application-Association-Identifiers, which need only be unambiguous within the scope of the cooperating AE-invocations, and thus do not have to be registered.

In connection-mode communications, the AET can be used in called, calling and responding application address parameters in A-ASSOCIATE (and RT-OPEN) service primitives. The ACSE service provides for the optional specification of an AET value by its component values (AP-title and AE-qualifier) in A-ASSOCIATE primitives. Similarly, the RTSE service (which itself makes use of ACSE) provides for the optional specification of an AET value by its component values in RT-OPEN primitives.

The calling/called AP title identifies the AP that contains the requester/acceptor of the A-ASSOCIATE service. The AE qualifier identifies the particular AE of the AP that contains the requester or the acceptor of the A-ASSOCIATE service. The AP and AE Invocation-identifiers identify the AP invocation and AE invocation that contain the requester or the acceptor of the A-ASSOCIATE service. The presence of each of these addressing parameters is defined in ISO standards as a user option.

Other information objects which must be named are the Application Context name and Presentation Context identifier which are exchanged at connect-time.

3.5.4 Addressing

According to ISO 7498-3 (Basic Reference Model - Naming and Addressing Addendum), a Presentation Address comprises a globally unique NSAP address, plus local Transport selector, Session selector and Presentation selector. The Session selector and Presentatoin aelector are not utilized in CNS/ATM-1 Package.

The only mandatory addressing parameters in the A-ASSOCIATE (and RT-OPEN) service primitives are the calling, called and responding Presentation Addresses. These are passed transparently by ACSE (and RTSE) to the Presentation Service.

3.5.5 Name - Address Mapping

Any OSI layer entity or application process can be named via a title. This must be translated into an address by means of a directory function either at Application or Network Layer. Each AE is attached to one or more PSAP and hence the AET is associated with the corresponding Presentation Address(es). The AET is mapped onto a Presentation Address by means of an Application Layer directory function

explained later in this document. The Application Layer directory function provides a mapping from an AET into the PSAP address required to access the referenced application entity.
The use of selectors is a local function and there may in practice be a direct correspondence between application entity titles and TSAP address or NSAP address.

3.5.6 Selectors in the ATN

Session service users are addressed at a Session Service Access Point (SSAP), where a session address consists of two parts - a session selector (Ssel) and a transport address. Likewise presentation service users are addressed at a Presentation Service Access Point (PSAP), where a presentation address consists of a presentation selector (Psel) and a session address. The various components are simple strings of zero or more octets and are meaningful only at a particular network address. An Application Process (AP) address can be defined as:

AP Address = NSAP Address + Tsel + Ssel (0) + Psel (0)

It follows that the values chosen for the Presentation and Session selectors is a local matter, although the actual values chosen must be made known to any potential ‘remote’ users - even if they are null. In the case of null presentation and session selectors the value of the TSAP can be used to convey information, such as the ATN application to be used.

For ATN upper layer addressing, the use of Session and Presentation selectors is not recommended. The use of Transport selectors is discussed in the section on the Transport Layer.
Upper layer addressing in ATN shall use null Session and Presentation selectors.

3.5.7 Registration Issues

It has been agreed that the ISO 9834 registration scheme should be put forward for adoption by ICAO.

A number of situations have been identified where object identifiers (OIDs) are being interchanged; some of these are registered elsewhere, some will need registration by ICAO. A given object should only be assigned one OID, ie. it should only be registered once (either by ICAO or by some other organization).

ICAO is in the process of setting up a registration authority under ISO. ICAO Working Groups will need to register information objects including ASOs, ASEs, Application Titles, Presentation Contexts through such a registration authority. It may also be necessary to set up a registration authority for Distinguished Names, as used by the Directory service and by systems management.

IATA, in developing the AOP, has adopted the ICD (or international) form of network address (NSAP) using ICD 0027, which is jointly administered by ICAO and IATA. Traditionally, IATA has assigned globally unique identifiers to international

organisations, which they use for both store-and-forward and transactional communications based on IATA standard protocols. These identifiers include, among others, the two-character airline codes assigned to IATA members and other organisations.

4. Implementation Issues

4.1 Introduction and Scope

This section examines the protocol overhead of various upper layer profiles. The term "upper-layer OSI protocols" is itself ambiguous. Sometimes it used to mean ACSE, Presentation and Session (sometimes referred to as A/P/S) - the protocols that are used directly or indirectly by application-layer protocols (the fact that ACSE is itself nominally in the application-layer adds to this confusion). Sometimes it means everything above the transport layer, including the application layer protocols. Most of the discussion in this section concerns the supporting upper-layers; at least some of it can be applied to application protocols proper. It draws on the analysis of [2.6].

There are so many options available in the different layers that a thorough and complete analysis is not possible in a document of this size. Assumptions have had to be made in order to simplify the analysis. These assumptions are based on three premises:

- The functions in the stack will be selected in order to optimise the use of the bandwidth available.

- The majority of the overhead will be due to association set up, release and data transfer. Cases where the association is not accepted and cases where other upper layer services are used are therefore not considered.

- As the transport and network overheads are also significant, the aim is to minimise the number of PDUs transferred; therefore, when the protocol allows it, user data will be piggy backed on top of other PDUs.

4.2 Evaluation of OSI Upper Layer protocol overheads

4.2.1 Basis of Overhead Analysis

The use of supporting upper layers by a real application may be specified in a profile, rather than explicitly by the application base standard. For example FTAM [ISO 8571] has an associated profile specifying the use of ACSE, Presentation and Session [ISP 10607-1]. Such profiles may specify that non-mandatory items in the

ACSE, Presentation and Session base standards are made mandatory for conformance to the profile. However, in this section the assumption is that the requirements of the application are based on the conformance clauses of these base standards.

The estimates below optionally include the most basic of upper layer addressing in the form of a nil selector for Session and Presentation selector addresses.

For the purposes of this document a very simple OSI Application is considered which will use ACSE, Presentation and Session at their most basic.

A detailed evaluation of the overheads incurred by use of the OSI upper layer protocols is presented in this chapter. In making this analysis, the following assumptions are made:

4.2.2 Basic Assumptions

a) "ACSE" refers to the second edition of ISO 8650 (1995). "A2CSE" refers to the DIS text to extend ACSE to support ASO-associations.

b) Session Version 2 is used.

c) Unless otherwise stated, all OPTIONAL items are omitted from PDU size calculations.

4.2.3 Encoding Options

d) When the ASN.1 Basic Encoding Rules (BER) are used for ACSE and Presentation headers, the length field is mostly the definite variant (short or long) option.

e) ASN.1 Packed Encoding Rules (PER) in this paper refers to the Basic Unaligned variant of ISO/IEC 8825-2. By imposing limits (bounds) on value ranges in the abstract syntax, use can be made of constrained and semi-constrained whole number coding (for lengths and integers).

f) ASN.1 type EXTERNAL is taken to be as defined in ISO/IEC 8824-1 (1992).

g) When a SEQUENCE OF SEQUENCE is encoded by ASN.1/PER it is assumed that the lower bound is zero and the upper bound is 255. In this way just one octet is required to specify the component count.

4.2.4 Name and Address Options

h) Where object identifiers (OIDs) are used, it is assumed that they are constrained such that no more than five arcs will be present and the component values will be small. This enables the OID to be encoded in 4 octets, with a length field of 1-octet

(for BER) or 2-bits (for PER). The OID values assumed are listed in section 5 of Annex C.

i) An AP-title is assumed to be an OID and an AE-qualifier is assumed to be a small value Integer (less than 128). The effect of including addressing fields in Called and Calling Identification fields in ACSE connect PDUs is illustrated in section 6 of Annex C.

j) Both the Presentation Selector and Session Selector have a value of NULL..

4.2.5 Presentation Context

k) When A2CSE is used, even the presentation-context-definition-list in Presentation CP and CPA PDUs are omitted 2 (A2CSE provides this information in its own PDUs).

l) It is assumed that in the case of BER encoding the default presentation context definition will use an abstract syntax of ACSE (Version 2) and a transfer syntax of ASN.1/BER. Basic communication applications do not require default Presentation Context in CP and CPA [CULR-3].

m) It is assumed that Presentation Context Identifiers are small valued integers (less than 128). One presentation context only is specified, this is for ACSE. If the application needs to specify a distinct abstract syntax, then the addition of this adds 15 octets (when selecting ASN.1/BER) or 17 octets (when selecting ASN.1/PER), as illustrated in Annex C.

n) Only one transfer syntax (ASN.1/BER or ASN.1/PER as appropriate) is negotiated.

4.2.6 User Data

o) Presentation user data is simply-encoded-data, that is an OCTET STRING.

p) The effect of application user data is shown as an option. User data is sent as an OCTET STRING and so its structure will be the subject of bilateral agreement.

5. PDU Definitions and Supporting ASN.1 Types

For convenience, the ASN.1 definitions of important ACSE, Presentation and Session PDUs used for the this overhead calculation are collected together in this section and annotated.

[2] Strictly the presence of the presentation-context-definition-list field is mandatory, as ISO 8823 Section 6.2.2.7 states "This shall be a list containing one or more items."

5.1 ACSE Definitions

The elements new to the ACSE edition 2 with the ASN.1extensibility notation (ISO/IEC 8650-1:1995,ed.2/PDAM2) of the connection-oriented ACSE are indicated by redlining.

```
ACSE-apdu :: = CHOICE
{
aarq        AARQ-apdu,          -- ACSE associate request pdu
aare        AARE-apdu,          -- ACSE associate response pdu
rlrq        RLRQ-apdu,          -- ACSE release request pdu
rlre        RLRE-apdu,          -- ACSE release response pdu
abrt        ABRT-apdu,          -- ACSE abort pdu
. ..
}
```

```
AARQ-apdu ::= [APPLICATION 0] IMPLICIT SEQUENCE
{
protocol-version                [0]  IMPLICIT BIT STRING
                                     {version1(0)} DEFAULT {version1 },
application-context-name        [1]  Application-context-name,
called-AP-title                 [2]  AP-title OPTIONAL,
called-AE-qualifier             [3]  AE-qualifier OPTIONAL,
called-AP-invocation-identifier [4]  AP-invocation-identifier OPTIONAL,
called-AE-invocation-identifier [5]  AE-invocation-identifier OPTIONAL,
calling-AP-title                [6]  AP-title OPTIONAL,
calling-AE-qualifier            [7]  AE-qualifier OPTIONAL,
calling-AP-invocation-identifier [8] AP-invocation-identifier OPTIONAL,
calling-AE-invocation-identifier [9] AE-invocation-identifier OPTIONAL,
-- The following field shall not be present if only the Kernel is used.
sender-acse-requirements        [10] IMPLICIT ACSE-requirements OPTIONAL,
-- The following field shall only be present if the Authentication FU is selected.
mechanism-name                  [11] IMPLICIT Mechanism-name OPTIONAL,
-- The following field shall only be present if the Authentication FU is selected.
calling-authentication-value    [12] EXPLICIT Authentication-value OPTIONAL,
application-context-name-list   [13] IMPLICIT Application-context-name-list
                                     OPTIONAL,
implementation-information      [29] IMPLICIT Implementation-data OPTIONAL,
extensions                           SEQUENCE {...} OPTIONAL,
user-information                [30] IMPLICIT Association-information OPTIONAL
}
```

```
AARE-apdu ::= [APPLICATION 1] IMPLICIT SEQUENCE
{
protocol-version                    [0]  IMPLICIT BIT STRING
                                         {version1(0),} DEFAULT { version1 },
application-context-name            [1]  Application-context-name,
result                              [2]  Associate-result,
result-source-diagnostic            [3]  Associate-source-diagnostic,
responding-AP-title                 [4]  AP-title OPTIONAL,
responding-AE-qualifier             [5]  AE-qualifier OPTIONAL,
responding-AP-invocation-identifier [6]  AP-invocation-identifier OPTIONAL,
responding-AE-invocation-identifier [7]  AE-invocation-identifier OPTIONAL,
-- The following field shall not be present if only the Kernel is used.
responder-acse-requirements         [8]IMPLICIT ACSE-requirements
                                         OPTIONAL,
-- The following field shall only be present if the Authentication functional unit is
      selected.
mechanism-name                      [9]  IMPLICIT Mechanism-name OPTIONAL,
-- The following field shall only be present if the Authentication functional unit is
      selected.
responding-authentication-value     [10]EXPLICIT Authentication-value OPTIONAL,
application-context-name-list       [11] IMPLICIT Application-context-name-list
                                                  OPTIONAL,
-- The above field shall only be present if the Application Context Negotiation
                                         functional unit is selected
implementation-information          [29] IMPLICIT Implementation-data OPTIONAL,
extensions                               SEQUENCE {...} OPTIONAL,
user-information                    [30] IMPLICIT Association-information OPTIONAL,
}
```

```
RLRQ-apdu ::= [APPLICATION 2] IMPLICIT SEQUENCE
{
reason                              [0]  IMPLICIT Release-request-reason OPTIONAL,
extensions                               SEQUENCE {...} OPTIONAL,
user-information                    [30] IMPLICIT Association-information OPTIONAL,
}
```

```
RLRE-apdu ::= [APPLICATION 3] IMPLICIT SEQUENCE
{
reason                              [0]  IMPLICIT Release-request-reason OPTIONAL,
extensions                               SEQUENCE {...} OPTIONAL,
user-information                    [30] IMPLICIT Association-information OPTIONAL,
}
```

-- For the purposes of this and associated papers, assume that an AE-Title is defined as:

AE-Title ::= OBJECT IDENTIFIER -- 7 arcs of low values {(1) (3) (27) a b c d}

5.2 Presentation Layer Definitions

5.2.1 Presentation PCI

The PPDUs defined for the short-encoding and the null-encoding options of the presentation protocol are indicated by redlining.

```
CP-type ::= SET                 -- Presentation Connect PDU
{
[0]         IMPLICIT Mode-selector,
[1]         IMPLICIT SET {COMPONENTS OF Reliable-Transfer-
              APDUs.RTORQapdu} OPTIONAL,
[2]         IMPLICIT SEQUENCE
            {
            [0]     IMPLICIT Protocol-version DEFAULT {version-1},
            [1]     IMPLICIT Calling-presentation-selector OPTIONAL,
            [2]     IMPLICIT Called-presentation-selector OPTIONAL,
            [4]     IMPLICIT Presentation-context-definition-list OPTIONAL,
            [6]     IMPLICIT Default-context-name OPTIONAL,
            [8]     IMPLICIT Presentation-requirements OPTIONAL,
            [9]     IMPLICIT User-session-requirements OPTIONAL,
                    User-data OPTIONAL
            } OPTIONAL
}
```

SHORT-CP PDU

The PCI of the SHORT-CP is one octet, with the two trailing bits consisting of the encoding-choice parameter. This PCI is followed by the User-data parameter (encoded as per the encoding-choice parameter, cf next section). The encoding of the SHORT-CP is as shown in the following bit pattern:

0000 00zz

where zz identifies the encoding choice as follows:
00: bilateral agreement
01: BER
10: unaligned PER
11: aligned PER

```
CPA-PPDU ::= SET                -- Presentation Connect Accept PDU
{
[0]  IMPLICIT Mode-selector,
[1]  IMPLICIT SET {COMPONENTS OF Reliable-Transfer-APDUs.RTOACapdu}
OPTIONAL,
[2]  IMPLICIT SEQUENCE
            {
            [0]     IMPLICIT Protocol-version DEFAULT {version-1},
```

```
            [3]     IMPLICIT Responding-presentation-selector OPTIONAL,
            [5]     IMPLICIT Presentation-context-definition-result-list OPTIONAL,
            [8]     IMPLICIT Presentation-requirements OPTIONAL,
            [9]     IMPLICIT User-session-requirements OPTIONAL,
                    User-data OPTIONAL
            } OPTIONAL
}
```

SHORT-CPA PDU

The PCI of the SHORT-CPA is one octet, with the two trailing bits consisting of the encoding-choice parameter. This PCI is followed by the User-data parameter (encoded as per the encoding-choice parameter, cf next section). The encoding of the SHORT-CP is as shown in the following bit pattern:

0000 00zz

where zz identifies the encoding choice as follows:
00: bilateral agreement
01: BER
10: unaligned PER
11: aligned PER

5.2.2 Presentation User Data

The encoding of the presentation User data required by the CNS/ATM-1 ULA is redlined.

```
User-data ::= CHOICE
{
        [APPLICATION 0] IMPLICIT Simply-encoded-data,
        [APPLICATION 1] IMPLICIT Fully-encoded-data
}

Simply-encoded-data ::= OCTET STRING

Fully-encoded-data ::= SEQUENCE
                                SIZE (1,...)
                                OF PDV-list

PDV-list ::= SEQUENCE
            {       Transfer-syntax-name OPTIONAL,
                    Presentation-context-identifier,
                    presentation-data-values  CHOICE {
                        single-ASN.1-type    [0] ANY,
                        octet-aligned    [1] IMPLICIT OCTET STRING,
                        arbitrary        [2] IMPLICIT BIT STRING
                        }
          }
```

The Transfer-syntax-name field shall not be present in the encoded presentation User Data.
The ”arbitrary” choice for presentation-data-value shall be used in the encoded presentation User Data.
The values of the Presentation-context-identifier are predefined as follows: 0 (acse-apdu), 1 (reserved for future use), 2(user-ase-apdu), other (reserved for future use).

```
Presentation-context-definition-list ::= Context-list

Context-list ::= SEQUENCE OF SEQUENCE
{
            Presentation-context-identifier,
            Abstract-syntax-name,
            SEQUENCE OF Transfer-syntax-name
}

Presentation-context-identifier ::= INTEGER
                                        (1..127,...)
```

```
Presentation-context-definition-result-list ::= Result-list

Result-list ::= SEQUENCE OF SEQUENCE
{
```

```
            [0] IMPLICIT Result,                    -- INTEGER(0..2)
                    -- Transfer-syntax-name shall be present if Result is "acceptance"
            [1] IMPLICIT Transfer-syntax-name OPTIONAL,
                    -- provider-reason shall be present if Result is "provider-rejection"
            provider-reason    [2] IMPLICIT INTEGER
                            {
                            reason-not-specified (0),
                            abstract-syntax-not-supported (1),
                            proposed-transfer-syntaxes-not-supported (2),
                            local-limit-on-DCS-exceeded (3)
                            } OPTIONAL
}

Result ::= INTEGER {
                acceptance (0),
                user-rejection (1),
                provider-rejection (2)
        }
```

5.3 Session Layer Definitions

The PPDUs defined for the short-encoding and the null-encoding options of the session protocol are indicated by redlining.

CONNECT (CN) SPDU SI = 13						
PGI	**m/nm**	**Code**	**PI**	**m/nm**	**Code**	**Octets**
Connection identifier	nm	1	Calling SS-user reference	nm	10	64 max
			Common Reference	nm	11	64 max
			Additional reference information	nm	12	4 max
Connect / Accept	nm	5	Protocol options	m	19	1
item			TSDU maximum size	nm	21	4
			Version number	m	22	1
			Initial serial number	nm	23	6 max
			Token setting item	nm	26	1
			Session user requirements	nm	20	2
			Calling session selector	nm	51	16 max
			Called session selector	nm	52	16 max
User Data	nm	193				512 max
			Data overflow	nm	60	1
Extended user data	nm	194				10240 max

SHORT CONNECT [SCN) SPDU						
	m/nm	**Octet**	**Parameter**	**m/nm**	**Value**	**Bir number**
SI&P	m	1	SI	m	101101	4-8
			Parameter indication	m	0	3

DATA TRANSFER (DT) SPDU SI = 1						
PGI	**m/nm**	**Code**	**PI**	**m/nm**	**Code**	**Octets**
			Enclosure item	nm	25	1
User Information Field						unlimited

TYPED DATA TRANSFER (TD) SPDU SI = 33						
PGI	**m/nm**	**Code**	**PI**	**m/nm**	**Code**	**Octets**
			Enclosure item	nm	25	1
User Information Field						unlimited

ACCEPT (CNA) SPDU SI = 14						
PGI	**m/nm**	**Code**	**PI**	**m/nm**	**Code**	**Octets**
Connection identifier	nm	1	Called SS-user reference	nm	9	64 max
			Common Reference	nm	11	64 max
			Additional reference information	nm	12	4 max
Connect / Accept item	nm	5	Protocol options	m	19	1
			TSDU maximum size	nm	21	4
			Version number	m	22	1
			Initial serial number	nm	23	6 max
			Token setting item	nm	26	1
			Token item	nm	16	1
			Session user requirements	nm	20	2
			Calling session selector	nm	51	16 max
			Responding session selector	nm	52	16 max
			Enclosure item	nm	25	1
User Data	nm	193				

SHORT ACCEPT (SAC) SPDU						
	m/nm	**Octet**	**Parameter**	**m/nm**	**Value**	**Bir number**
SI&P	m	1	SI	m	111101	4-8
			Parameter indication	m	0	3

FINISH (FN) SPDU SI = 9						
PGI	**m/nm**	**Code**	**PI**	**m/nm**	**Code**	**Octets**
			Transport disconnect	nm	17	1
			Enclosure item	nm	25	1
User Data	nm	193				

DISCONNECT (DN) SPDU SI = 10						
PGI	**m/nm**	**Code**	**PI**	**m/nm**	**Code**	**Octets**
			Enclosure item	nm	25	1
User Data	nm	193				

5.4 ASN.1 EXTERNAL Type Definition

ASN.1 UNIVERSAL type EXTERNAL is taken to be as defined in ISO DIS 8824-1(1992):

```
EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE
{
        direct-reference        OBJECT IDENTIFIER OPTIONAL,
        indirect-reference      INTEGER OPTIONAL,
        data-value-descriptor       ObjectDescriptor OPTIONAL,
        encoding CHOICE
        {
                single-ASN1-type        [0] ANY,
                octet-aligned           [1] IMPLICIT OCTET STRING,
                arbitrary           [2] IMPLICIT BIT STRING
        }
}

ObjectDescriptor ::= [UNIVERSAL 7] GraphicString
```

5.5 Definition of Object Identifiers

Entity	**Object Identifier**	**BER Encoding**
ATN-App (application context)	{iso icd(3) icao(27) atn-ac(3) 0 (0)}	0x 2B 1B 03 00
ACSE	{joint-iso-ccitt association-control(2) abstract-syntax(1) apdus(0) version1(1)}	0x 52 01 00 01
ASN.1/BER	{joint-iso-ccitt asn1(1) basic-encoding(1)}	0x 51 01
ASN.1/PER	{joint-iso-ccitt asn1(1) packed-encoding(3) basic(0) unaligned(1)}	0x 51 03 00 01

6. Connection Oriented Upper Layer Overheads

Overhead estimates for connection oriented upper layers (ACSE, Presentation and Session) are presented in this section. Two estimates are given, as follows:

- ACSE ed.1/BER + full Presentation/BER + full Session (standard stack)
- ACSE ed.2/PER + Presentation ”Fast Byte” + Session ”Fast Byte” (ULA stack)

6.1 Connection Establishment (Request)

The association initialization will be carried by the ACSE A-ASSOCIATE request [AARQ] which in turn is carried by the Presentation P-CONNECT request [CP] which in turn is carried by the Session S-CONNECT. It is assumed that there is no ATN-App user data during the association establishment phase.

6.1.1 ACSE Association Request [AARQ]

The only parameter assumed to be required in the AARQ APDU is the ASO-context-name. There is no user data (overhead generated by user data is discussed in section 3.1).

ASN.1/BER encoding of ACSE ed.1 AARQ

```
------
-- ASN.1/BER Encoding of ACSE Ed. 1 AARQ-apdu
------
T = 0x60                                    -- Application 0 [AARQ apdu]
L = 0x??
        T = 0xA1                            -- application-context-name
        L = 0x06
                T = 0x06                    -- Object Identifier
              L = 0x04
                V = 0x2B 0x1B 0x03 0x00     -- {Ctx 0}
```

With no user data, the AARQ overhead is thus 10 octets for ACSE.

ASN.1/PER encoding of ACSE ed. 2 AARQ

```
-----
--  ASN.1/PER Encoding of ACSE Ed. 2 AARQ-apdu
-----

Extensibility          =    B`0’
ACSE-apdu.Choice =    B`000'
Preamble               =         B`00000000 00000001'          -- only user-info present
ASO-context-name =          B`11'                          -- no of arcs less 1
                                0x2B 0x1B 0x03 0x00          -- {ctx 0}
```

With no user data, the AARQ overhead is thus 6 octets and 6 bits.

That APDU has to be embedded in a Fully-encoded-data structure in order to allow the ATN-App CF to distinguish APDUs when there are several abstract syntaxes defined.

```
-----
--  ASN.1/PER Encoding of the CF PDV
-----

Number of PDVs     =     0x01                          --  PDV-list
Preamble           =          B`0'                    -- transfer-syntax-name absent
presentation-ctx-id =         0x00                  --  presentation-context-identification
p-d-v              =          B`10'                  -- choice = arbitrary
```

The encoding of the CF PDV generates an overhead of 2 octets + 3 bits. The actual AARQ overhead is thus 9 octets + 1 bit.

6.1.2 Presentation Connect Request [CP]

The following parameter values are assumed:

- a Mode-selector of normal,
- Calling-presentation-selector (Optional) not used,
- Called-presentation-selector (Optional) not used,
- Presentation-context-definition-list:
  - Abstract syntax of ACSE ed.1
  - Application layer's abstract syntax ATN-App
  - Transfer syntax of either ASN.1/BER or ASN.1/PER
- simply-encoded user-data (ie. an OCTET STRING).

ASN.1/BER Encoding of CP

```
-----
-- Full Presentation : CONNECT PDU (CN)
-----
T = 0x31                                        -- Presentation CP-type
L = 0x??
        T = 0xA0                                -- mode-selector
        L = 0x03
        V = 0x80 0x01 0x00                      -- normal
        T = 0xA4                                -- Presentation-context-definition-list
        L = 0x??
                T = 0x30                        -- SEQUENCE OF SEQUENCE {}
                L = 0x??
                        T = 0x02                -- Presentation-context-identifier
                        L = 0x01
                        V = 0x01                -- PCI = 1
                        T = 0x06                -- Object Identifier
                        L = 0x04
                        V = 0x52 0x01 0x00 0x00 -- ACSE
                        T = 0x30                -- SEQUENCE OF Transfer-syntax-name
                        L = 0x04
                                T = 0x06 -- Object Identifier
                                L = 0x02
                                V = 0x51 0x01   -- ASN.1/BER
                        T = 0x02                -- Presentation-context-identifier
                        L = 0x01
                        V = 0x02                -- PCI = 2
                        T = 0x06                -- Object Identifier
                        L = 0x04
                        V = 0x?? 0x?? 0x?? 0x?? -- ATN App
                        T = 0x30                -- SEQUENCE OF Transfer-syntax-name
                        L = 0x04
                                T = 0x06 -- Object Identifier
                                L = 0x02
                                V = 0x51 0x03 0x00 0x01         -- ASN.1/PER
        -- user-information (See section 5.2)
```

With no user data, the CP overhead is thus 43 octets.

Encoding of the SHORT CP

```
-----
-- Short-encoding option ("Fast Byte") : SHORT CONNECT PDU (SCP)
-----
PCI = B`000000'                         -- Presentation Short CP
zz  = B`10'                             -- transfer syntax = PER
        -- user-information (See section 5.2)
```

With no user data, the SCP overhead is thus 1 octets.

6.1.3 Session Connect Request

If Session User Requirements is absent, the default is as though half-duplex, minor synchronization, activity management, capability data and exceptions functional units had been selected. However, by specifying a value of 0x?? 0x?? duplex functional unit is selected.

```
SI = 0x0D                                   -- CN-SPDU
LI = 0x??
        PGI = 0x05                          -- Connect/Accept Item
        LI = 0x06
            PI = 0x13                       -- Protocol Options
            LI = 0x01
            PF = 0x00
            PI = 0x16                       -- Version
            LI = 0x01
            PF = 0x02                       -- Version 2
            PI = 0x14                       -- User Requirements
            LI = 0x02
            PF = 0x0? 0x0?                      -- See text
        PGI = 0xC1 | C2                         -- User Data Preamble
        LI = 0x?? | 0xFF 0x?? 0x??
```

The Basic Session-CN Header Size is thus 14 octets. If the Session User data is not greater than 512 octets, then the data header adds another 2 octets, otherwise 4 octets are added, giving a header size of:

- 16 octets (short user data)
- 18 octets (long user data)

Encoding of the SHORT CONNECT PDU (SCN)

```
-----
-- Short-encoding option ("Fast Byte") : SHORT CONNECT PDU (SCN)
-----
SI = B`101101'                          -- Session Short CN
P  = B`0'                       -- no parameter
        -- user-information (See section 5.2)
```

With no user data, the SCN overhead is thus 1 octets.

6.2 Connection Establishment (Response)

6.2.1 ACSE Association Response [AARE]

The only parameters assumed to be required are protocol version, application-context-name, result, result-source-diagnostic. The application receives no user-information.

ASN.1/BER Encoding of ACSE Ed. 1 AARE

```
-----
-- ASN.1/BER Encoding of ACSE Ed 1. AARE-apdu
-----
T = 0x61                                        -- AARE-apdu
L = 0x??
        T = 0xA1                                -- application-context-name
        L = 0x06
            T = 0x06                            -- OBJECT IDENTIFIER
            L = 0x04
            V = 0x2B 0x1B 0x30 0x00             -- { Ctx 0}
        T = 0xA2                                -- result
        L = 0x03
            T = 0x02                            --
            L = 0x01
            V = 0x??
        T = 0xA3                                -- result-source-diagnostic
        L = 0x05
            T = 0xA0 | 0xA1                             --
            L = 0x03
                    T = 0x02                    -- Integer
                    L = 0x01                    -- Bounded 0..10
                V = 0x??
```

With no user data, the AARE overhead is thus 22 octets for ACSE

ASN.1/PER Encoding of ACSE Ed. 2 AARE

```
-----
-- ASN.1/PER Encoding for ACSE Ed. 2 AARE-apdu
-----
Extensibility bit                   B`0'
ACSE.apdu.Choice =                  B`001'
Preamble  =                             B`00000000 001'         -- user-info only
Application-context-name =              B`11'                   -- no of arcs
                                        0x2B 0x1B 0x03 0x00     -- {Ctx 0}
Associate-result =                      0x??                -- INTEGER (assume 0..255)
result-source-diagnostic =              B`?'                    -- CHOICE of 2
                                        0x??                -- INTEGER (assume 0..255)
```

With no user data, the AARE overhead is thus 8 octets + 2 bits for ACSE. In addition to the CF PDV encoding, the AARE overhead is therefore 10 octets + 6 bits.

6.2.2 Presentation Connection Response [CPA]

The following parameter values are assumed;

- a Mode-selector of normal,
- acceptance of the presentation context definition and,
- simply encoded user-data.

ASN.1/BER Encoding of CPA

```
-----
-- Full Presentation: ACCEPT CONNECT PDU (CPA)
-----
T = 0x31                                        -- CPA-PPDU
L = 0x??
        T = 0xA0                                -- mode-selector
        L = 0x03
                T = 0x80
                L = 0x01
                  V = 0x01                      -- normal
        T = 0xA2                                -- [2] SEQUENCE
        L = 0x??
                  T = 0xA5                      -- Presentation-context-definition-result-list
                L = 0x??
                          T = 0x30                      -- SEQUENCE OF SEQUENCE
                        L = 0x??
                                  T = 0x80              -- [0] Result
                              L = 0x01
                                  V = 0x??              -- acceptance
                                  T = 0x81              -- [1] Transfer-syntax-name
                              L = 0x02
                                  V = 0x51 0x03 0x00 0x01       -- {ASN.1/PER}
                                  T = 0x80              -- [0] Result
                              L = 0x01
                                  V = 0x??              -- acceptance
                                  T = 0x81              -- [1] Transfer-syntax-name
                              L = 0x02
                                  V = 0x51 0x03 0x00 0x01       -- {ASN.1/PER}
```

With no user data, the CPA overhead is thus 31 octets for ACSE Ed. 1.

Encoding of the SHORT CPA

```
-----
-- Short-encoding option ("Fast Byte"): SHORT CONNECT ACCEPT PDU (SHORT-CPA)
-----
PCI     = B`000000'                             -- CPA-PPDU
zz      = B`10'                                     -- transfer syntax = PER
```

With no user data, the SHORT-CPA overhead is thus 1 octet.

6.2.3 Session Connection Accept [CNA]

The session user requirements are used to indicate the functional units proposed by the Session Connection Request (see previous calculation for CN-SPDU).

```
SI = 0x0E                                   -- CNA-SPDU
LI = 0x??
        PGI = 0x05                          -- Connect/Accept item
        LI = 0x06
                PI = 0x13                   -- Protocol Options
                LI = 0x01
                PF = 0x00
                PI = 0x16                   -- Version Number
                LI = 0x01
                PF = 0x02
        PI = 0x14                           -- Session User Requirements
        LI = 0x02
        PF = 0x00 0x02                              -- See Text
        PGI = 0xC1 | 0xC2                   -- User Data
        LI = 0x?? | 0xFF 0x?? 0x??
        PF = xx xx ... xx xx                -- PPDU
```

The Basic Session-CNA Header Size is 14 octets. If the Session User data is not greater than 512 octets, then the data header adds another 2 octets; otherwise 4 octets are added, giving a header size of:

- 16 octets (short user data)
- 18 octets (long user data)

Encoding of the SHORT CONNECT ACCEPT PDU (SCA)

```
-----
-- Short-encoding option (”Fast Byte”) : SHORT CONNECT PDU (SCA)
-----
SI = B`101101'                          -- Session Short Accept
P  = B`0'                       -- no parameter
```

With no user data, the SCA overhead is thus 1 octets.

6.3 Connection Termination (Request)

6.3.1 ACSE Release Request [RLRQ]

This can potentially have no content.

ASN.1/BER Encoding of ACSE Ed.1 RLRQ

```
T = 0x62
L = 0x00
```

The ACSE Ed. 1 RLRQ overhead is thus 2 octets.

ASN.1/PER Encoding of ACSE Ed.2 RLRQ

```
ACSE-apdu.Choice =      B‘010’
Preamble =              B‘000’
```

The ACSE Ed. 2 RLRQ overhead is thus 6 bits. Including the CF-PDV encoding, the actual overhead is 3 octets + 1 bit.

6.3.2 Presentation Release Request

The P-RELEASE presentation service is carried by a User-Data PPDU. Since the defined context set (DCS) cannot change over the lifetime of the presentation connection (the context management functional unit is not selected), User-data can be conveyed as simply-encoded-data (ie. an Octet String). As the A-RELEASE encoding is small in size, use short-definite length encoding.

ASN.1/BER Encoding of Presentation Release Request

```
T = 0x60                                -- User-Data PPDU
L = 0x??
V = xx xx ... xx xx                     -- ACSE RLRQ
```

When selecting the null-option of the Presentation protocol, no overhead is generated by the presentation during the connection release phase.

6.3.3 Session Finish [FN]

To close the session down, use a FINISH SPDU and send this on T-DATA:

```
SI = 0x09                               -- FN-SPDU
LI = 0x00
CN = none
```

When selecting the null-option of the Session protocol, no overhead is generated by the session during the connection release phase.

6.4 Connection Termination (Response)

6.4.1 ACSE Release Request [RLRE]

This can potentially have no content.

ASN.1/BER Encoding of ACSE Ed.1 RLRE

```
T = 0x63
L = 0x00
```

The ACSE Ed. 1 RLRQ overhead is thus 2 octets.

ASN.1/PER Encoding of ACSE Ed. 2 RLRE

```
ACSE-apdu.Choice =      B‘011’
Preamble =              B‘000’
```

The ACSE Ed. 2 RLRQ overhead is thus 6 bits. Including the CF-PDV encoding, the actual overhead is 3 octets + 1 bit.

6.4.2 Presentation Release Response

The P-RELEASE presentation service is carried by a User-Data PPDU. Since the defined context set (DCS) cannot change over the lifetime of the presentation connection (no context management functional unit), convey User-data as simply-encoded-data (ie. an Octet String). As the A-RELEASE encoding is small in size, use short-definite length encoding.

ASN.1/BER Encoding of Presentation Release Response

```
T = 0x60                                    -- User-Data PPDU
L = 0x??
V = xx xx ... xx xx
```

When selecting the null-option of the Presentation protocol, no overhead is generated by the presentation during the connection release phase.

6.4.3 Session Disconnect [DN]

The response to a Session FINISH is a DISCONNECT:

SI = 0x0A -- DN-SPDU
LI = 0x00
CN = none

When selecting the null-option of the Session protocol, no overhead is generated by the session during the connection release phase.

6.5 Data Transfer Phase

Application data is mapped onto P-DATA, so there is no ACSE involvement here.

The overhead generated by the CF PDV encoding is 2 octets + 3 bits.

6.5.1 Presentation

The structure of the data carried is a matter for bilateral agreement, it being delivered to the application as an anonymous octet string.

ASN.1/BER Encoding of Presentation Data

There are a number of ways of encoding the Presentation data using BER, including the following:

a) For up to 127 octets of data, use the short variant of definite length encoding, giving an overhead of 2 octets:

T = 0x60 -- simply-encoded-data
L = 0x?? -- Length £ 127 octets
V = xx xx ... xx xx

b) For up to 65,535 octets of data, use the long variant of definite length encoding with a length of lengths of 2. This results in an overhead of 4 octets:

T = 0x60 -- simply-encoded-data
L = 0x82 0x?? 0x??
V = xx xx ... xx xx

c) For unlimited data lengths, use Indefinite length encoding. This results in an overhead of 4 octets:

T = 0x60 -- simply-encoded-data
L = 0x80
V = xx xx ... xx xx
¶ = 0x00 0x00

Alternatives (b) and (c) have the same overhead, but alternative (c) is easier to implement. Indefinite length encoding of data is therefore the preferred method.

When selecting the null-option of the Presentation protocol, no overhead is generated by the presentation during the data transfer phase.

6.5.2 Session

The Session Data Transfer Phase has a number of options;

a) normal data transfer, using the S-DATA service. In order to support basic concatenation, the Give Token SPDU is transferred with each (normal) Data SPDU

b) typed data transfer, using the S-TYPED-DATA service. This service is normally used to convey protocol control information outside of the normal data stream. The Typed Data functional unit needs to be active (and specified in the S-CONNECT)

c) expedited data transfer, using the S-EXPEDITED-DATA service. An Expedited SPDU can carry a maximum of 14 octets of user information.

Typed Data gives an overhead of 2 octets:

```
SI = 0x21                          -- TD-SPDU
LI = 0x00
-- user information
```

Normal data transfer gives an overhead of 4 octets:

```
-- GT-SPDU (S-GIVE-TOKEN required for basic concatenation)
SI = 0x01
LI = 0x00
-- DT-SPDU (S-DATA)
SI = 0x01
LI = 0x00
-- Followed by user-information (ie the P-DATA PPDU)
```

Expedited data transfer gives an overhead of 2 octets:

```
SI = 0x05                          -- EX-SPDU
LI = 0x00
```

When selecting the null-option of the Session protocol, no overhead is generated by the session during the data transfer phase.

7. User-Data and User-Information

7.1 Association-information (ASN.1/BER Encoded)

There are a number of ways of encoding ACSE Association-information using Basic Encoding Rules, including:

a) If the maximum length of Association-information is 123 octets, then the following short-definite length form of encoding can be used, giving an overhead of 6 octets.

```
T = 0xBE                                -- Association Information
L = 0x??                                        -- Length £ 127 octets
        T = 0x08                        -- EXTERNAL
        L = 0x??                                -- Length £ 125 octets
                T = 0x81                -- OCTET STRING
                L = 0x??                        -- Length £ 123 octets
                        V = xx xx ... xx xx     -- user information
```

b) If a generalized case is required, with no restriction on the length of Association-information, then indefinite length encoding must be used. This method results in an overhead of 12 octets.

```
T = 0xBE                                -- Association Information
L = 0x80
        T = 0x08                        -- EXTERNAL
        L = 0x80
                T = 0x81                -- OCTET STRING
                L = 0x80
                        V = xx xx ... xx xx     -- user information
                ¶ = 0x00 0x00
        ¶ = 0x00 0x00
¶ = 0x00 0x00
```

7.2 Presentation User-data (ASN.1/BER Encoded)

There are a number of ways of encoding Presentation User-data using BER. If we restrict the choice to simply-encoded-data the following methods can be used:

a) For up to 127 octets of data, use the short variant of definite length encoding. This results in an overhead of 2 octets:

```
T = 0xA0
L = 0x??
V = xx xx ... xx xx
```

b) For up to 65,535 octets of data, use the long variant of definite length encoding with a length of lengths of 2. This results in an overhead of 4 octets:

```
T = 0xA0
L = 0x82 0x?? 0x??
V = xx xx ... xx xx
```

c) For unlimited data lengths, use indefinite length encoding, giving an overhead of 4 octets:

```
T = 0xA0
L = 0x80
V = xx xx ... xx xx
¶ = 0x00 0x00
```

7.3 Association-information (ASN.1/PER Encoded)

There are a number of ways of encoding ACSE Association-information using PER:

```
Preamble =      B`0001'          -- encoding only choice in EXTERNAL
Preamble =      B`010'       -- OCTET STRING
Length =        B`0'             -- Length less than 128
Length =        B`???????'       -- Length
User-data =   xx xx ... xx  xx
```

a) For up to 127 octets of data:

b) For up to 16,384 octets of data:

```
Preamble =      B‘0001’          -- encoding only choice in EXTERNAL
Preamble =      B‘010’       -- OCTET STRING
Length =        B‘10’            -- up to 16K
Length =        B‘?????? ????????’
User-data =     xx xx ... xx xx
```

7.4 Presentation User-data (ASN.1/PER Encoded)

If the choice is restricted to simply-encoded-data the following method can be used:

```
Length =        B‘10’            -- Length < 16K
Length =        B‘?????? ????????’
User-data =     xx xx ... xx xx
```

8. ADDITIONAL OPTIONS

8.1 Effect of Basic Addressing Information

An AP-title is assumed to be an OID and an AE-qualifier is assumed to be a small value Integer (less than 128). The effect of including these addressing fields in Called and Calling Identification fields in ACSE connect pdus is as follows:

```
-- ASN.1/BER Encoding of ACSE Called/Calling ID fields
T = 0xA?                                          -- AP Title [2] or [6]
L = 0x06
        T = 0x08                                  -- OBJECT IDENTIFIER
        L = 0x04
        V = 0x?? 0x?? 0x?? 0x??
T = 0xA?                                          -- AE Qualifier [3] or [7]
L = 0x03
        T = 0x02                                  -- INTEGER
        L = 0x01
        V = 0x??                                  -- Assume £ 127
T = 0xA?                                          -- AP Invocation Id [4] or [8]
L = 0x03
        T = 0x02                                  -- INTEGER
        L = 0x01
        V = 0x??                                  -- Assume £ 127
T = 0xA?                                          -- AE Invocation Id [5] or [9]
L = 0x03
        T = 0x02                                  -- INTEGER
        L = 0x01
        V = 0x??                                  -- Assume £ 127

-------------------------------------------------------------------------
-- ASN.1/PER Encoding of ACSE Called/Calling ID fields
Length = B`11'                                    -- Length of OID less 1
Value = B`???????? ???????? ???????? ????????'   -- OBJECT IDENTIFIER
Value = B`????????'                               -- Assume £ 127
Value = B`????????'                               -- Assume £ 127
Value = B`????????'                               -- Assume £ 127
```

8.2 Effect of Additional Abstract Syntaxes

If the application needs to specify a distinct abstract syntax, then the addition of this adds 15 octets (when selecting ASN.1/BER) or 17 octets (when selecting ASN.1/PER), as follows:

```
-- Each additional abstract syntax (to be included in the
-- SEQUENCE OF SEQUENCE within the presentation-context-
-- definition-list) will encode as:
T = 0x02                                -- P-ctx-identifier
L = 0x01
V = 0x??
T = 0x06                                -- Abstract-Syntax
L = 0x04
V = 0x?? 0x?? 0x?? 0x??                 -- Five arc OID
T = 0x10                                -- SEQUENCE OF Transfer-syntax-name
L = 0x04
        T = 0x06
        L = 0x02
        V = 0x51 0x01                   -- {ASN.1/BER}

----
-- PER encoding
----
0x??                                     -- INTEGER (assume 0..255)
B‘11’                                    -- no of OID arcs less 1
0x?? 0x?? 0x?? 0x??                      -- Abstract Syntax
0x01                                     -- No of Transfer syntaxes
B‘11’                                    -- no of OID arcs less 1
0x51 0x03 0x00 0x01                      -- {ASN.1/PER}
```

9. SUMMARY

<Editor’s note: this section shall be reviewed>

The overhead due to upper layers depends upon the inclusion or exclusion of user-information and its ultimate encoding. The estimates are based on a number of assumptions which are detailed in section 3.2 in the body of the Standing Document.

The ASN.1/BER used in producing these estimates does not give a unique octet sequence during encoding. Different designs of encoder will produce different, but correct, encodings (cf. Distinguished Encoding Rules, which guarantee that an encoding can only be performed one way - thus encodings can be `string compared'). If minimum size is the ultimate goal, then PER should be examined. However, PER is not currently allowed in the base standards for ACSE and Presentation.

As an example, if an application generates an A-ASSOCIATE with no user-data, then transmits 1024 octets of data, and then terminates, the overheads incurred are as summarised in the following sections.

Options:

- *The addition of called and calling address information to ACSE adds 23 octets to the ASN.1/BER encoding or 7¼ to the ASN.1/PER encodings.*

- *If Presentation is expected to carry between 126 and 65,535 octets of data this adds 4 octets to the P-DATA.*

9.1 Overhead Using a standard stack and an ULA stack

	ACSE ed.1 + full Presentation + full Session	CF PDV + ACSE ed. 2 + short and null encoding option of Presentation and Session (octets + bits)
Establish Connection - Request CF PDV ACSE AARQ Presentation Session	0 10 43 16	2 + 3 6 + 6 1 1
S	69	11 + 1
Establish Connection - Response CF PDV ACSE AARE Presentation Session	0 22 31 16	2 + 3 8 + 2 1 1
S	69	10 + 7
Data Transfer Phase CF PDV Presentation P-DATA Session S-TYPED-DATA	0 4 2	2 + 3 0 0
S	6	2 + 3
Terminate Connection - Request CF PDV ACSE RLRQ Presentation P-RELEASE Session FN	0 2 2 2	2 + 3 0 + 6 0 0
S	6	3 + 1
Terminate Connection - Response CF PDV ACSE RLRE Presentation P-RELEASE Session DN	0 2 2 2	2 + 3 0 + 6 0 0
S	6	3 + 1
TOTALS	156	30 + 5

10. CNS/ATM-1 Package Guidance Material

10.1 Introduction

10.1.1 Motivation of the Work

10.1.1.1 Introduction

The original discussion for the CNS/ATM-1 ULA is presented.

The section discusses the proposed ULA in terms of its relation to two other proposals. These proposals are the application-over-transport proposal, and the current OSI solution.

As requirements for an ATN ULA are extracted, they are stated as Rn statements. After the two current approaches are discussed, the ATN ULA is presented. The Rn statements are then collated and ULA satisfaction of those requirements are stated in complementary Un statements.

10.1.1.2 Applications over transport solution

Previous work, is specified directly over the transport layer. This leads naturally to the question. Why should one have a ULA at all? If one can send messages over a transport provider, what else is required?

As one reads such a specification, certain characteristics of that application design process naturally emerge. For example, message processing in terms of message identification, message urgency and message response characteristics are all specified. Clearly, the application must handle duplicate messages. Each application needs to perform this function separately. Second, message urgency and response requires the application to implement an urgency scheme to handle multiple messages correctly.

As one reads further, the full encoding of each message in terms of Abstract Syntax Notation 1 (ASN.1) and Packed Encoding Rules (PER) is specified. Clearly, without a ULA, an application over transport must itself fully specify the complete encoding of each message.

The first point, then, is that examination of previous specifications has ULA requirements. However, as each application is constructed, it must itself build and specify each application element that it requires, even if that application element exists in a standardized form, or exists elsewhere. Upper layer requirements exist. The choice is a use of a standardized ULA and associated tools, or specification and implementation of upper layer elements individually for each application. Reusability of existing work is an important goal. This leads to shorter development, smaller code, and certification credit.

R1. ULA Requirements (message accountability, encoding) exist for all cited applications, regardless of whether they are formalized in a ULA.

R2. Common application service elements are an important ULA contribution

The application requires an orderly termination of the application dialogue. A transport service alone cannot perform an orderly termination. If a data (DT) transport protocol data unit (TPDU) is sent, and then a disconnect request (DR) TPDU is sent, clearly the delivery of the DT is not guaranteed. Indeed, a DT can have been acknowledged by the Transport Service Provider (TS-Provider), and still not delivered to the Transport Service User (TS-User) if the DR arrives. Clearly, this requirement must thus be handled above transport.

The definition of when message exchange begins and ends can only be defined by the cooperating applications. If this is not done in a current application, the transport layer cannot help. The orderly termination of the message exchange is a requirement for every current application. This message exchange is generally referred to as an association. It is crucial to the discussion to realize that an application over transport makes the association synonymous with the transport connection.

R3. Every application has a requirement for association control (orderly initiation and termination of message exchange)

An important consequence of this is that every application must open a transport connection. Thus, for example, the Context Management (CM) dialogue consists of exactly one uplink and one downlink. Still, an entire transport connection setup and teardown is required to support the one exchange. Clearly, aircraft running multiple applications to one air traffic control (ATC) center would benefit immensely from multiplexed associations over a single transport connection. The conservation of bandwidth by multiplexing aircraft-center applications is a worthwhile benefit.

R4. Multiple applications running over a single transport connection is an important goal.

The security requirements of the initial ATN applications merit consideration. The hole in initial authentication requirements for previous applications is well-known. For example, the TWDL application implements extensive procedures for transition to the Next Data Authority (NDA), but the aircraft simply responds to the first Connection Request (CR) it receives as its initial data authority. Security requirements are not specified in current application-over-transport standards.

R5. There exist CNS/ATM-2 requirements authentication security requirements for each application.

The previous hole might be addressed as follows. If it were required that CM executed before CPDLC, then there is a start to a guarantee that the entity calling the aircraft actually got the address from CM.

R6. Currently specified applications have implicit requirements for application control in terms of application-to-application interaction and application ordering.

This section has considered certain design aspects of applications over transport. Certain ULA aspects present in these applications were discussed and certain ATN ULA requirements were discussed.

10.1.1.3 Classic OSI solution

The ATN standards have consistently stated that the ATN is fully based on International Organization for Standardization (ISO) standards for Open System Interconnection (OSI). For the ATN to succeed, international voluntary standards must be followed, rather than proprietary specifications.

R7. The ATN is an OSI standard architecture.

ISO has specified a seven-layer reference model. At the upper layers, application, presentation, and session protocols are specified. Certain aspects of the protocol are not workable for the ATN. For example, a full upper layer protocol establishment requires 98 octets. This is not workable for air-ground subnetworks.

R8. Current OSI Upper Layer protocol overhead is unacceptable for air-ground links.

It is also clear no current ATN air-ground applications require any of the services offered by the session protocol.

R9. There are no current air-ground requirements for the OSI session protocol (ISO 8327, edition 2) functional units.

The ISO presentation protocol similarly has greater functionality than required. For example, there is no need for negotiation of complete defined context sets (DCS) for ATN applications.

R10. The OSI presentation protocol (ISO 8823, edition 2) must be tailored for ATN use.

This short discussion indicates that the classic OSI protocols are untenable in terms of protocol overhead and also in terms of extra protocol functionality.

10.1.1.4 The ATN ULA

The ATN ULA has three aspects. These are as follows:

ISO 9545, edition 2 specifies the extended application layer structure (ALS). This allows modular construction of software by specification of application service elements (ASEs). An ASE is implemented as a software module. These are combined into Application Service Objects (ASOs). Interaction between ASEs and ASOs are mediated by a control function (CF).

ISO 8649, edition 2 and ISO 8650, edition 2 specify the Application Control Service Element (ACSE) needed to support the ALS. ACSE allows the establishment of associations over transport connections.

Amendments to ISO 8823 and ISO 8327 specify efficient Presentation Protocols and Session Protocols. The amendments specify minimal functionality protocols to indicate protocol functionality.

The ATN ULA uses the ATN transport service.

10.1.1.5 Review of ULA requirements conformance

The requirements for the ULA are listed below.

R1. ULA Requirements (message accountability, encoding) exist for all cited applications, regardless of whether they are formalized in a ULA.

U1. The ULA offers a formal method of ASO specification. This also promotes software reusability. Certification authorities have reacted positively to the use of ALS for certification credit, such that ASOs do not have to be recertified.

R2. Common application service elements are an important ULA contribution.

U2. In the ULA, ASEs can be formally specified. They can be formally checked for correctness. They can then be combined in a modular fashion.

R3. Every application has a requirement for association control (orderly initiation and termination of message exchange).

U3. The association control service element (ACSE) offers a tool for association control for every application.

R4. Multiple applications running over a single transport connection is an important goal.

U4. ACSE directly supports the higher level associations. All extensions to ACSE are bit-wise backwards compatible with previous editions of ACSE.

R5. There are CNS/ATM-2 authentication security requirements for each application.

U5. ACSE has an authentication element which is an important element in a security architecture.

R6. Currently specified applications have requirements for application control in terms of application-to-application interaction and application ordering.

U6. The use of the CF allows precise specification of application interaction and application ordering.

R7. The ATN is an OSI standard architecture.

U7. When the ATN SARPs are published, every ATN ULA element will be an OSI standard. Every ULA enhancement is consistent with the OSI architecture.

R8. Current OSI Upper Layer protocol overhead is unacceptable for air-ground links.

U8. The upper layer efficiency standards reduce the upper layer establishment overhead from 98 octets to one.

R9. There are no current air-ground requirements for the OSI session protocol (ISO 8327, edition 2) functional units.

U9. The session functional units are not in the ATN ULA. If and when required, the session functional units will be recast.

R10. The OSI presentation protocol (ISO 8823, edition 2) must be tailored for ATN use.

U10. The OSI efficiency measures do not require the use of the current presentation protocol.

10.1.2 Architectural Guidance Material

10.1.2.1 Description of the Upper Layer Architecture

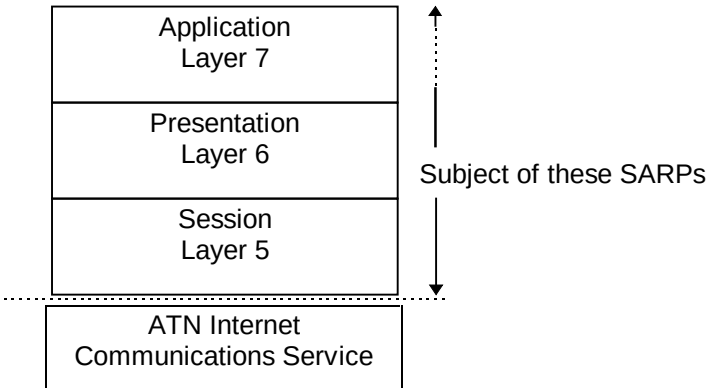


Figure 10.1: Conceptual view of the scope of the UL SARPs

Note -- In the picture above, it is useful to note the effect of the supporting layer efficiency enhancements.
First, the connect request at the application layer is mapped to the connect request of the supporting upper layers.
Second, the data request at the application layer is mapped directly to the transport data request.
Third, there is no congruent mapping for the release request. It must be handled by the Control Function using data requests (to carry the ACSE release) and abort requests (to actually clear the connection after the graceful release.)

10.1.2.2 Description of the Application Layer

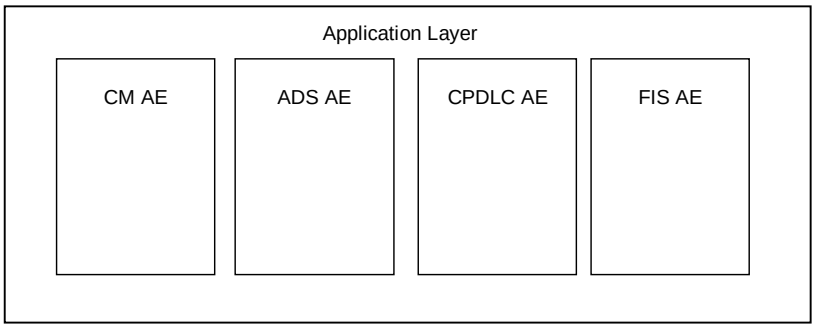


Figure 10.2: Conceptual view of Application Layer

Note -- It is important to note the architectural decisions embodied in the above picture.
Each application is embodied in an AE.
Each AE contains an ATN ASE, which is the communications element responsible for an ATN application. The internal structure of the ATN ASE may be of arbitrary complexity. The dialogue service is the ATN ASE's view of the ATN ULA.
Thus, the type of the AE is the same as the type of the ASE. That is, an ADS AE will contain only an ADS ASE. Thus, on connection establishment, the peer title can be completely constructed. The AE-title is provided by the peer ID value in the dialogue service, and the peer AP-qualifier (e.g., CPC) is the same as yours is.
There is no architectural capability for multiple instances of the same ATN ASE within the same AE. A requirement for another instance of an ASE (e.g., the CM contact procedure), requires spawning another AE. This implies that the ATN ASE will generate and manage only one dialogue over the lifetime of the ASE.
There is no interaction inside an AE between ASEs of different types, since these are not provided by the architecture. Any requirement for interaction between applications occurs in user space through, e.g., global data structures.
As implied in the figure, there is no upper-layer addressing. All addressing of the AE-type is complete with the TSAP address. All upper layer selectors are necessarily nil.
The application context name is also a simple construct, since the CNS/ATM-1 ASE list can be inferred from the AP-title. The CNS/ATM-1 application context name encodes only the application version.

10.1.2.3 Description of the AE Structure

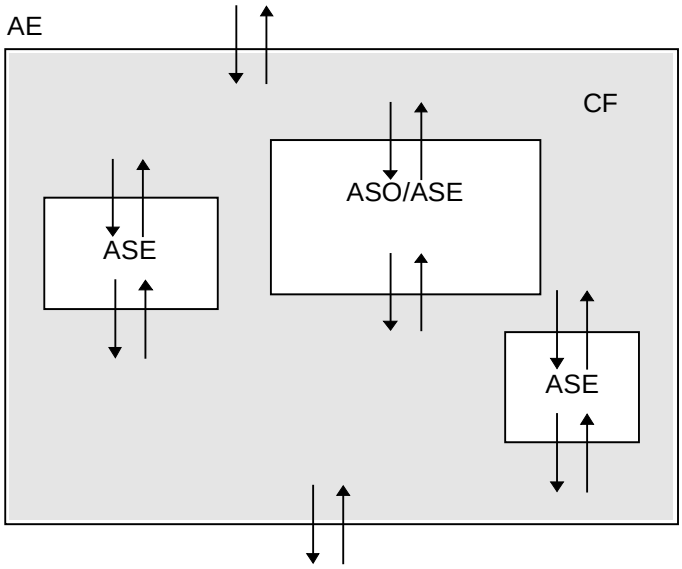


Figure 10.3: Generic Application Entity structure

The internal picture of the AE indicates that the AE comprises several ASEs. The AE picture is completed by the presence of an upper and lower service boundary. Each ASE picture is similar in form, with an upper and lower service boundary. The job of the Control Function is solely to map inputs and outputs to and from ASE and AE service boundaries. By definition, the CF cannot produce protocol; in that case, an ASE is required.
The confirmed data service element (CDSE) is still undergoing requirements analysis. The CDSE is shown as an example of the inclusion in the ATN ULA of a common ASE constructed for use by all ATN applications.
In CNS/ATM-1, the upper AE service boundary is identical with the upper ASE service boundary. In CNS/ATM-1 all AE inputs are mapped to the ATN ASE.

10.2 Dialogue Service

10.2.1 Introduction

The dialogue service is a service that allows a user to bind to an association, send data, and unbind from the association.

10.2.2 Connection Mode

The dialogue service provides a connection mode upper layer service to the application. There is no connectionless mode upper layer architecture in CNS/ATM-1. Thus, there is no use of the connectionless transport protocol in CNS/ATM-1.

10.2.3 Mapping to transport

The ATN ULA makes extensive use of the user-data capability of the transport service. The ULA attempts to map the AARQ (containing user data) to the T-CONNECT. If this is not possible, based on user-data size, the T-CONNECT is sent, and the T-DATA is used to convey the AARQ (including user data). The implementor is advised to consult the PDU calculations in the present Guidance Material, but generally if the DS-User places more than five octets of user-data in the D-START, the D-START is mapped to the T-CONNECT + T-DATA.

The details of the mapping of D-START values (i.e., Residual Error Rate (RER) and Traffic Type) to transport values is for urgent resolution.

10.2.4 Control Function (Air-Ground)

The air-ground Control Function (CF) mapping function is described in the following figure. The five threads to implement the CF are described.

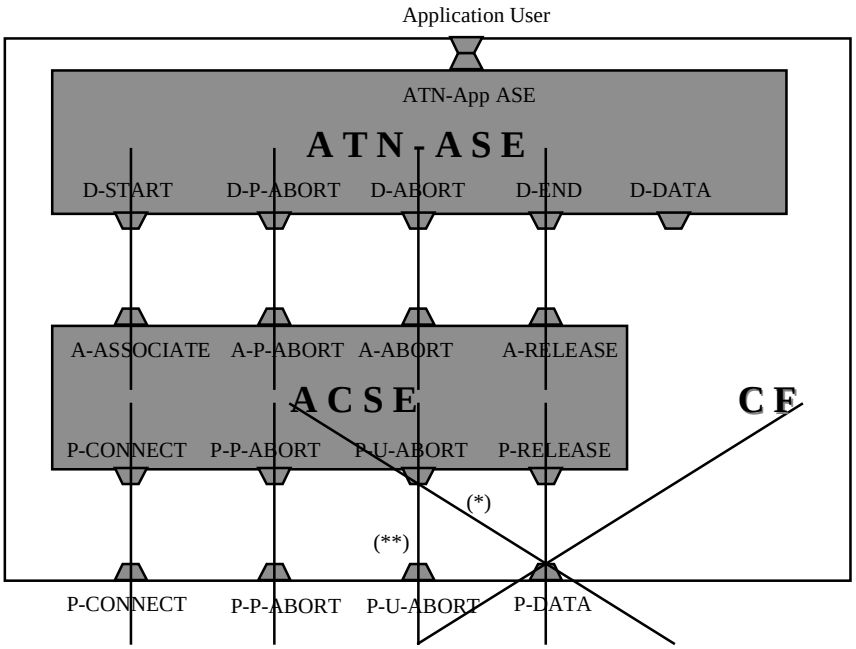
The D-START is mapped to the A-ASSOCIATE, which is mapped to the P-CONNECT by ACSE. The P-CONNECT is then mapped to the P-CONNECT (which is the T-CONNECT plus fast byte).

The D-P-ABORT occurs as an indication (‘up’) only. The P-P-ABORT is mapped to the A-P-ABORT. The A-P-ABORT is mapped to the D-P-ABORT.

The D-ABORT is mapped to the A-ABORT. The A-ABORT is mapped to the P-U-ABORT. The P-U-ABORT with user-data is mapped to the P-DATA which is followed by a P-U-ABORT. The P-U-ABORT without user-data is mapped to the P-U-ABORT.

The D-END is mapped to the A-RELEASE, which becomes a P-RELEASE. The P-RELEASE is mapped to the P-DATA.

The D-DATA is mapped to P-DATA, which in CNS/ATM-1 is syntactically equivalent to T-DATA.



(*) user data in the abort
(**) no user data in the abort

10.3 Session

10.3.1 Session Defect Report

If the AARQ fits in the T-CONreq, but the AARE does not fit in the T-CONcnf, the session state machine will interpret the lack of user-data in the T-CONcnf as a refusal of fast-byte capability. A continue bit should be defined in the session short accept, meaning that user-data follows the accept.

10.4 Presentation

10.4.1 Presentation Defect Report

When a D-STARTrsp- produces a P-CONrsp-, the presentation state machine has an empty transition. The transition (P-CONrsp-, Await S-P-CONrsp) should be [SCR, STAI0]

10.4.2 Presentation Fast Byte Guidance Material (OSIEFF)

Null functionality at a layer refers to the case where no functionality is required of a layer during the data transfer phase but where OSI compatibility and compliance are required. This is the case which is most clearly applicable to the ITU-T applications which use “short stacks”, to permit much greater OSI compatibility while at the same time allowing efficient communications. While it is possible to use the normal OSI layer protocol to signal that null functionality shall be required in the data phase, in certain instances, it is also possible to use a different protocol which is considerably more efficient (in terms of byte efficiency and, possibly, connection-establishment efficiency) to perform the negotiation. The term “fast byte” has been employed, as a convenient mnemonic, to refer to the insertion of a single byte PCI at connection establishment to signal that no further PCI will flow for that instance of communication. The use of the fast byte at a layer therefore serves to provide a service mapping between the layer above it and the layer below it.

NOTE — Early discussions on the fast byte concept considered the possibility of using the byte — theoretically only a bit — to serve as a *placeholder* so that it was possible, at some point during the instance of communication, to use the normal layer protocol to re-negotiate the use of some layer functionality where none was required earlier. At least for the upper layers, such a dynamic re-negotiation of layer functionality is for further study.

Thus, if a transport layer fast byte is exchanged, the layer service remains the same, i.e., the transport fast byte is a different version of the transport protocol with a one-to-one mapping of the network services to the transport services. In other words, by using the transport fast byte, one gets a QoS which is only as good as that provided by the underlying network service. The lower layer fast bytes are particularly useful for cases where the applications are communicating over a single data link, as is the case with ISDN access signalling.

For the upper layers, the typical, normal OSI implementation requires a 13 to 20 octet overhead on a single presentation data value (pdv) using the presentation and session data transfer services. This overhead is necessary to identify the state of the communication (i.e., that it is the data transfer phase as opposed to, say, the release phase), and to identify the pdv as belonging to a particular presentation context. Clearly, a null PCI optimization for the data phase implies a reduction in the layer service available to the application. For instance, in this case where all the application data is carried directly as user-data of the transport service, there is no guarantee that an encoded application pdu will not resemble a session spdu;

therefore, null PCI for the session data transfer phase implies that it is not possible to distinguish session spdus from application PCI. Therefore, it is not possible to use the orderly release facility of the session layer, though, of course, the application protocol can be defined to perform this function. Similarly, null PCI for presentation data transfer implies that there can only be one presentation context for the application pdus, whose abstract transfer syntaxes are known a priori. Thus, reducing the upper layer functionality inherent in the null functionality data phase restricts the range of applications that can use this optimization.

This loss of functionality must be reflected to the user at the service interface. For the session and presentation layers, the layer services are bundled together in groups known as functional units. At this time, orderly release of the session connection is provided as a part of the mandatory kernel functional unit. The use of null encoding for the data phase requires that the users have negotiated the use of a new functional unit, the so-called no orderly release functional unit, which removes the orderly release from the kernel functional unit.

NOTE — The orderly release capability would more logically be a functional unit separate from the kernel; the new "negative" functional unit provides compatibility with the current specificatons that require the (non-negotiable) kernel to be indivisible.

To this end, ITU-T Rec. X.pfbs defines the pass-through access to the session service, in particular the (new) no orderly release functional unit. As the presentation layer uses the session layer services for release of the presentation connection, there is no reduction to the presentation services. Thus efficiency optimizations available at the presentation layer are new protocol options, i.e., alternative, efficient PCI and procedures.

ITU-T Rec. X.pfbp define two protocol options at the presentation layer that greatly reduce the quantity of presentation PCI in cases where the presentation user's requirements for presentation functionality are limited. The null-encoding protocol option provides an alternative presentation protocol option for data transfer with zero PCI which can be negotiated at connection establishment only if one of the following conditions described below is true:

a) the presentation context definition list contains precisely one item in which the abstract syntax name is known to the responding presentation protocol machine by bilateral agreement; or

b) the presentation context definition list is empty and the default context is known by bilateral agreement; or

c) the presentation context definition list is empty and the abstract syntax of the default context is known to the responding presentation protocol machine by bilateral agreement and is specified in ASN.1.

NOTE — It may be possible in the future to negotiate the null-encoding protocol option for efficient data transfer using the presentation protocol defined in ITU-T Rec. X.226 (1994) | ISO/IEC 8823-1:1994. It is left for further study to define an alternative version of the presentation protocol encoded using PER which will permit byte-efficient presentation negotiation of the full set of presentation functionality.

In addition, it is possible to use another protocol option, the short-encoding option, which defines encodings for some presentation PPDUs which are considerably shorter than the current ones if *both* conditions d) and e) described below are true:

d) the calling and called presentation selectors are null; and

e) the presentation-requirements parameter in the P-CONNECT service includes the kernel functional unit only.

The short-encoding protocol option allows the negotiation of the encoding rule which shall be used as the transfer syntax of the application PCI belonging to the single presentation context (which may be the default context) from one of BER, the aligned

and unaligned variants of PER or a “transparent” encoding which is understood by bilateral agreement.

ITU-T Rec. X.sfbs specifies the no orderly release functional unit, whose selection by the session user indicates that the user has no requirements for orderly release of the session connection. Thus, either the application protocol has been chosen to perform this function, or the application association (which is one-to-one with the underlying session connection) is released by disconnecting the transport connection or by an abortive release of the session connection. The selection of this functional unit by the initiating session user permits the initiating session protocol machine to offer the use of the null-encoding protocol option on the established session connection. The responding session protocol machine can accept this option if the responding session user has selected only (and nothing other than) the kernel, full-duplex and no orderly release functional units for use on the connection.

ITU-T Rec. X.sfbp describes how the negotiation of the null encoding protocol option can be done using the protocol options field of the session establishment SPDUs defined in ITU-T Rec. X.225 (1995) | ISO/IEC 8327-1:_____. However, ITU-T Rec. X.sfbp also defines the possibility of using the short-encoding protocol option for the establishment SPDUs, which define a one byte PCI for these SPDUs which are distinct from the leading octet of the current SPDUs, which provides a byte-efficient negotiation of the null-encoding protocol option provided that there is no session layer addressing information required to be exchanged, i.e., the session selectors are null.

It is expected that the short-encoding protocol option will be used in conjunction with the transport connection set-up to achieve interworking with current implementations and, for the case where the responder also implements this protocol option, achieve an improvement in round-trip efficiency by setting up the upper layer connections concurrently with the transport connection. This is achieved as follows: the SHORT CONNECT SPDU — which is the short-encoding version of the current session CONNECT SPDU — is sent as user data of the T-CONNECT request service primitive. This requires that the SHORT CONNECT SPDU plus any accompanying user data meet the 32 octet limitation on the size of the transport user data.

NOTE — The transport protocol class 0 does not permit the carriage of user data. Therefore, for this scenario to work, the transport protocol clsss 4 should be available at both ends, or the transport fast byte protocol should be employed.

Current session implementations ignore any user data on the T-CONNECT indication primitive, or, at worst, disconnect the transport connection. Thus, absence of any user data on the T-CONNECT confirm primitive is a signal to the initiating session protocol machine that the responder is an implementation of the current standards. If the responding session entity implements the short-encoding protocol option, the SHORT ACCEPT SPDU is sent as user data of the T-CONNECT response service primitive, and its receipt by the initiating session protocol machine completes the session connection establishment in tandem with the transport connection establishment.

Of course, the short-encoding option may be used with the T-DATA service for the case where an already established transport connection is assigned to the session connection. Interworking is not fully achieveable as there is no guarantee that the responding session entity, if based on the current standards, will send a REFUSE SPDU to signal a protocol error, which is what a short-encoding for an SPDU would be.

10.5 ACSE

10.5.1 Discussion of differences in ACSE editions

As the ACSE is the major part of available software in the ULA implementation, a description is provided of the evolution of the ACSE standard. The editor's preface to ISO 8649, Service Description, was amended three times from the first edition to the second edition. The three amendments are 1) Peer-entity Authentication during Association Establishment, 2) Connectionless ACSE Service, and 3) Application Context Name Negotiation. There were also three technical corrigenda (Tcs) cited. The most important of the TCs resolved a defect wherein EXTERNAL events could affect ACSE sequencing and state machine. The EXTERNAL events were added to ACSE in a TC to answer a defect that pointed out that a session resynchronization could purge (destroy) the session finish or disconnect that the A-RELEASE is traveling on. Now, clearly over the ATN ULA of strict fast-byte supporting upper layers none of this matters, since there are no external events. A classic implementation, though, must put in a few EXTERNAL hooks at the bottom of its state machine when things go wrong in classic session/presentation.

ISO 8650-1, edition 2, has two Amendments and four TCs noted in the Editor's Preface. AM1 is Authentication, AM2 is Application Context Name negotiation, and TC2 is the 'EXTERNAL' TC.

The changes in the ACSE protocol specification are roughly similar, e.g., for TC2, "A-RELEASE procedure is disrupted if P-RESYNCHRONIZE, P-U-EXCEPTION-REPORT, or P-EXCEPTION-REPORT primitives occur on the association".

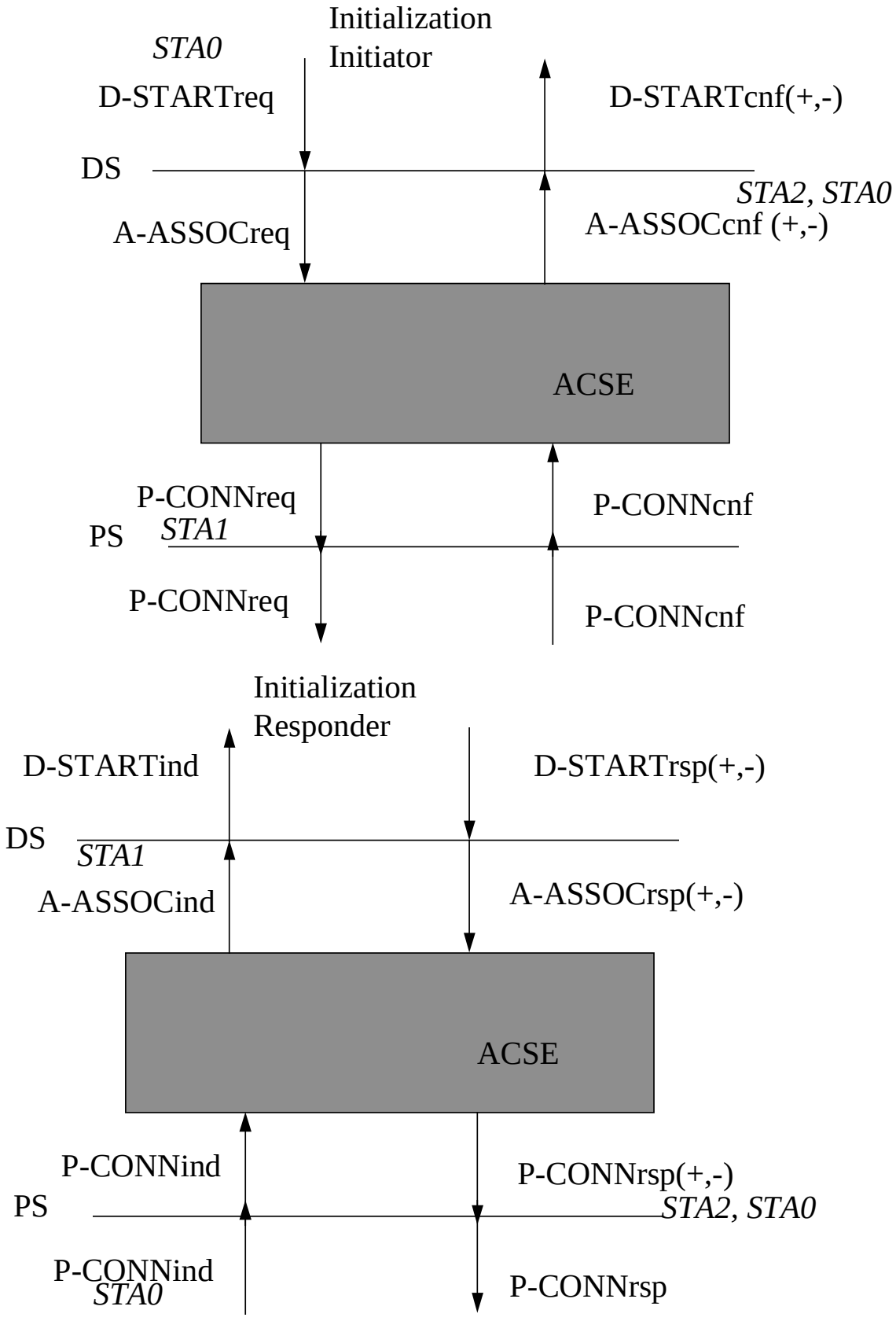
The state machine also adds two stimuli for EXTERN-1 and EXTERN-2. These stimuli cause the ACPM to return to the Associated state from one of the Attempting Release state.

The discussion indicates that the CNS/ATM-1 ULA requires none of the changes that distinguish ACSE, edition 1 from ACSE, edition 2.

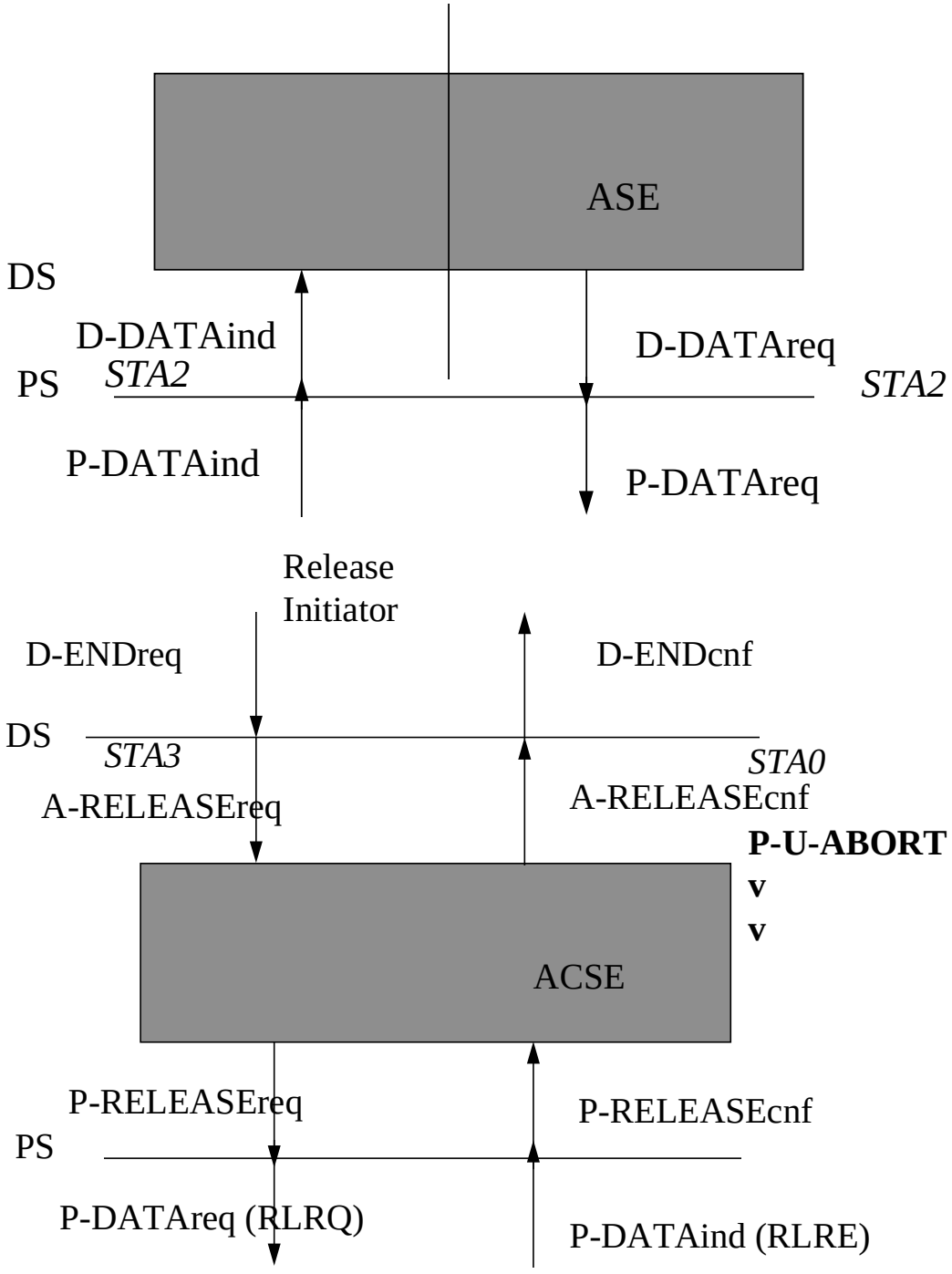
The two changes to ACSE that are required are the requirements to encode the ACSE PDUs in PER, and to map the P-RELEASE to P-DATA.

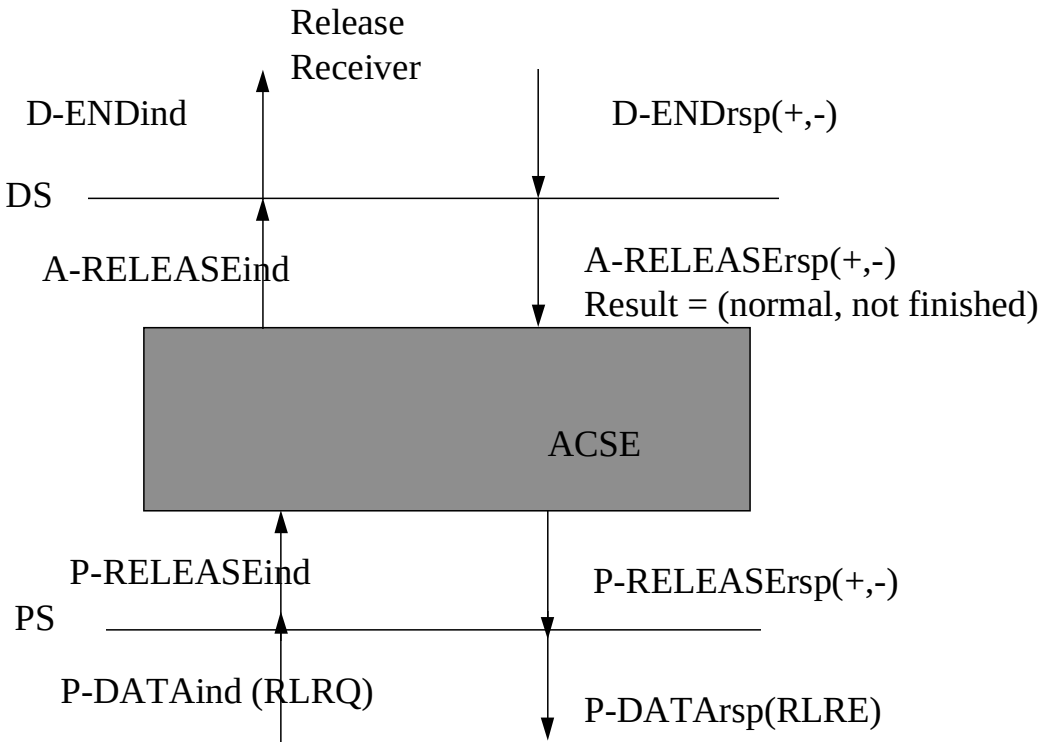
10.5.2 ACSE Primitive Flow Diagrams

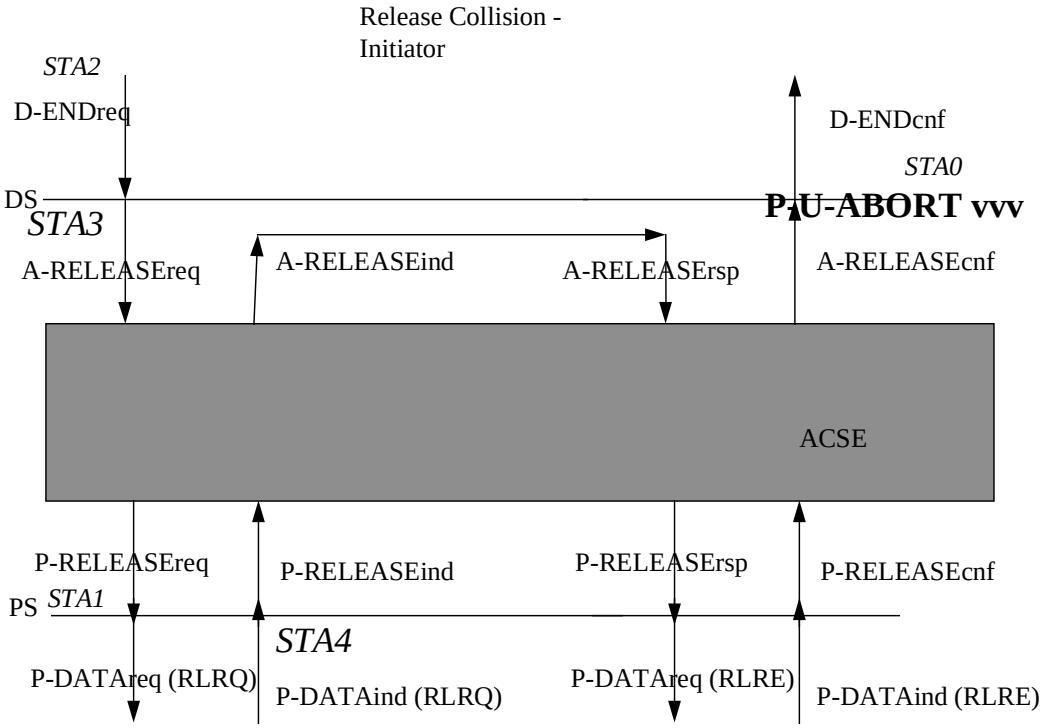
The role of the CNS/ATM-1 Control Function (CF) is to modulate the interaction of the ATN ASE and the ACSE. To that end, description of the characteristic flows involving ACSE is provided in the figures below.

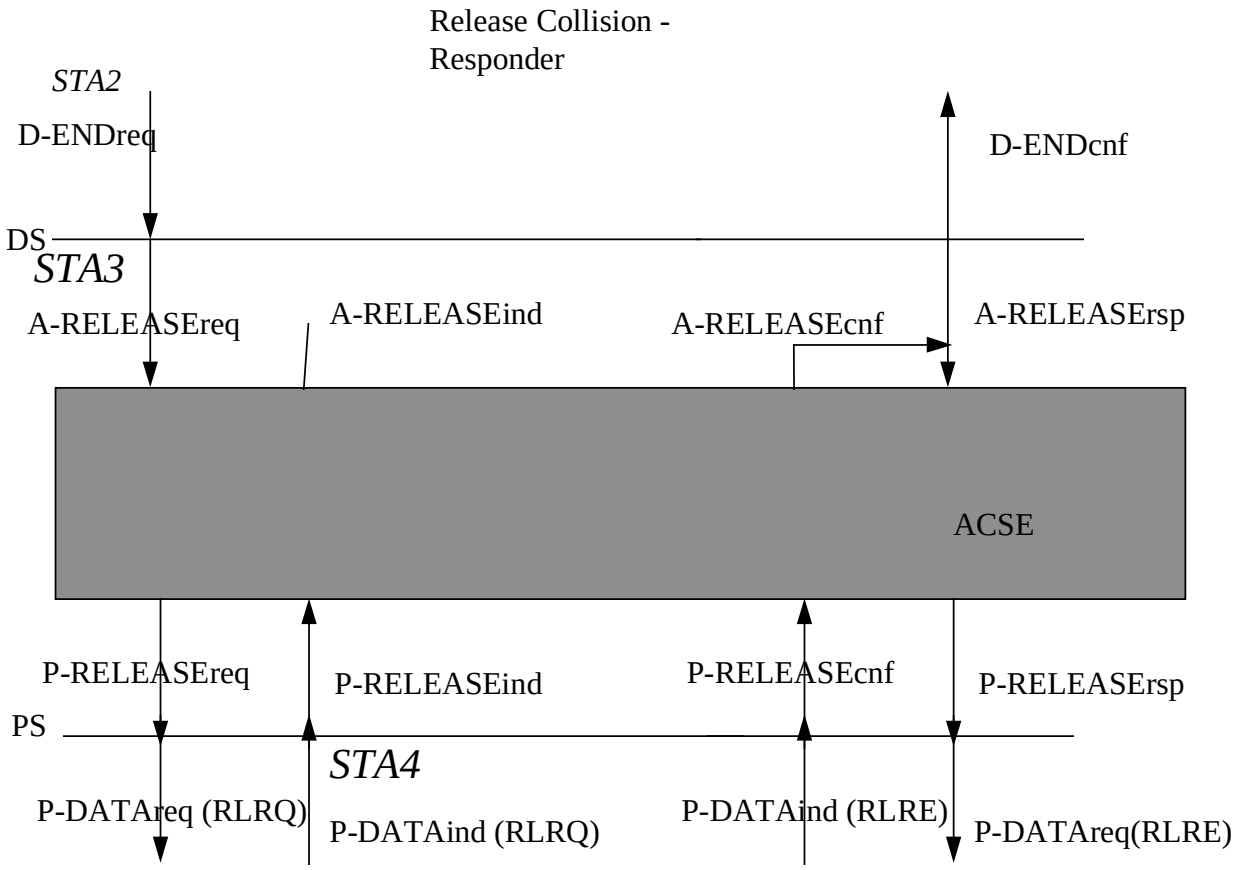

Initialization
Initiator
STA0
D-STARTreq
D-STARTcnf(+,-)
DS
STA2, STA0
A-ASSOCreq
A-ASSOCcnf (+,-)
ACSE
P-CONNreq
P-CONNcnf
PS
STA1
P-CONNreq
P-CONNcnf
Initialization
Responder
D-STARTind
D-STARTrsp(+,-)
DS
STA1
A-ASSOCind
A-ASSOCrsp(+,-)
ACSE
P-CONNind
P-CONNrsp(+,-)
PS
STA2, STA0
P-CONNind
STA0
P-CONNrsp

Data Transmission
Data Receipt




Release
Receiver
D-ENDind
D-ENDrsp(+,-)
DS
A-RELEASEind
A-RELEASErsp(+,-)
Result = (normal, not finished)
ACSE
P-RELEASEind
P-RELEASErsp(+,-)
PS
P-DATAind (RLRQ)
P-DATArsp(RLRE)


Release Collision -
Initiator
STA2
D-ENDreq
D-ENDcnf
STA0
DS
P-U-ABORT vvv
STA3
A-RELEASEreq
A-RELEASEind
A-RELEASErsp
A-RELEASEcnf
ACSE
P-RELEASEreq
P-RELEASEind
P-RELEASErsp
P-RELEASEcnf
PS
STA1
STA4
P-DATAreq (RLRQ)
P-DATAind (RLRQ)
P-DATAreq (RLRE)
P-DATAind (RLRE)


Release Collision -
Responder
STA2
D-ENDreq
D-ENDcnf
DS
STA3
A-RELEASEreq
A-RELEASEind
A-RELEASEcnf
A-RELEASErsp
ACSE
P-RELEASEreq
P-RELEASEind
P-RELEASEcnf
P-RELEASErsp
PS
STA4
P-DATAreq (RLRQ)
P-DATAind (RLRQ)
P-DATAind (RLRE)
P-DATAreq(RLRE)

10.6 Naming and Addressing

10.6.1 Implementation of ULA Construction of Titles and Addresses

The TSAP comprises the IDP (3 octets), the RDP (5 octets), the LDP (ARS (3) + 9 = 12 octets) and the SEL (0-2 octets) fields.

The fields are supplied as follows:
IDP = a priori
RDP = CMA (long TSAP - short TSAP) (Response)
ARS = D-START Called Peer ID (For air initiated requests only)
-- Note work is being done to determine what is required for the ground.
LDP - ARS + SEL = CMA short TSAP (Response)

The application name comprises the AE-title = AP-title + AE-qualifier

The fields are supplied as follows

AP-title is derived from the D-START Called Peer ID (Request)
AE-qualifier is derived from the CMA ASN.1 type APName

The CMA returns TSAP fragments and AE-titles (AP-titles + AE-qualifiers). The TSAP fragments are turned into complete PSAP addresses as follows:

a) If the TSAP fragment comprises only the local domain part (LDP), the Routing Domain Part (RDP) is restored from the previous TSAP fragment - name pair.

b) The Initial Domain Part (IDP) is restored to the prefix of the TSAP

c) No presentation or session selectors are added to the TSAP address.

This results in a complete application name - address mapping.

The received application context is mapped to the appropriate name and address by matching ATN ASE-type and presentation address.

11. SARPs Defect Register

1. No open DRs