ATN Upper Layers Guidance Material

ATNP/WG3/SG3, 15 October 1997

Version 5.1

REVISION STATUS

Version	Date	Comment
1.0	Feb 1996	First draft Guidance Material for Upper Layers (WG3 Gold Coast)
2.0	Apr 1996	Second draft Guidance Material for Upper Layers (WG3 Bruxelles)
3.0	Jun 1996	Third draft Guidance Material for Upper Layers (WG3 Munich)
4.0	Oct 1996	Fourth draft Guidance Material for Upper Layers (WG3 Alexandria)
4.1	May 1997	SG3 Annapolis
5.0	June 1997	Fifth draft Guidance Material for Upper Layers (WG3, Langen)
5.1	15/10/97	Update for WG3 Redondo Beach.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

1.1.1 This guidance material has been developed as a companion document to the ATN Upper Layer Communications Service (ULCS) standards and recommended practices (SARPs). It may be read alongside the ATN ULCS SARPs, in order to provide a greater understanding of the specification itself. Alternatively, readers who simply want to understand the purpose of the ULCS rather than the detail of the specification may read it instead of the ATN ULCS SARPs.

1.2 Scope

1.2.1 This document provides guidance material for ULCS SARPs. It does not define any mandatory or optional requirements, neither does it define any recommended practices.

1.2.2 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.

1.2.3 The basic aim is to define a set of architectural principles to allow ATN Applications to be specified and 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.

1.3 History

1.3.1 At its first meeting, the ICAO ATN Panel established a working group (WG3) to develop SARPs for the initial set of applications and ATN upper layers. The working group established a sub-group (SG3) to develop validated SARPs and guidance material for the ATN upper layer architecture.

1.3.2 The ULCS SARPs were validated through a number of methods, but chiefly through the development of independent implementations of the applications, and the successful running of interoperability trials between them.

1.4 Structure of Guidance Material

1.4.1 The guidance material closely follows the structure of the ULCS SARPs, so that for each area specified in the SARPs, guidance can be found under the same major section number in this material. In addition, there are three extra sections at the end of this material which go beyond the limited scope of the CNS/ATM-1 SARPs.

1.4.2 INTRODUCTION - contains the reason for providing guidance material as well as the scope. In addition, it provides a brief overview of upper layers functionality, the relationship with other SARPs, and identifies applicable reference documents.

1.4.3 DIALOGUE SERVICE - provides guidance and explanation of the abstract service which is referenced by application SARPs.

1.4.4 APPLICATION ENTITY -. provides guidance and explanation of the components which make up an AE in the CNS/ATM-1 package, including the role of the Control Function, and naming and addressing guidance.

1.4.5 SESSION - provides guidance and explanation of the "fast byte" session profile specified in the ULCS SARPs.

1.4.6 PRESENTATION - provides guidance and explanation of the "fast byte" presentation profile specified in the ULCS SARPs.

1.4.7 ACSE provides guidance and explanation of the ACSE profile specified in the ULCS SARPs.

1.4.8 EXAMPLE ENCODING - provides complete examples of the encoded data exchanged between users of the ATN Internet.

1.4.9 FUTURE MIGRATION - provides guidance and explanation of the intended future direction of the ATN Upper Layers, including a description of the provisions for forward compatibility in the present CNS/ATM-1 SARPs.

1.4.10 IMPLEMENTATION DECISIONS provides guidance on implementation-specific matters.

1.5 Definitions

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

1.5.2 *Application Entity (AE)*: an aspect of an application process pertinent to OSI, this 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 the ADS application as defined in ATN SARPs 2.

1.5.3 *Application Service Object (ASO)*: the generic term for an object which performs some communications related task. For example, the ADS protocol defined in 2.2.1.5 is an ASO specification. An AE is a particular kind of ASO, the "outermost" or highest level ASO. An ASO can contain further ASOs, which are thus recursively defined.

1.5.4 *Application Service Element (ASE)*: An AE can be broken down further into a number of ASEs, each of which provides 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 specified in separate standards. For example, the *Association Control Service Element (ACSE)* is used to create and release associations between applications. The applications in ATN SARPs 2 and 3 each define one or more ASEs.

1.5.5 *Control Function (CF)*: this exists within an ASO to co-ordinate the use of the different services provided in the constituent 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. This is specified in detail in 3.

1.5.6 *Invocations*. 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.

1.5.7 *Associations*. ASOs cannot co-operate until invoked. When two (or possibly more) ASOIs co-operate, 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. The peer ASOIs in an ASO association are not necessarily of the same type, but must be of complementary types. For example, if they are to exchange data, both must understand the same data syntax.

1.5.8 *Application Association*: a special type of ASO association which exists between an AE in one application and a peer AE in another. An application association underlies every other ASO association during the lifetime of the AEI. This is illustrated in Figure 1-1.

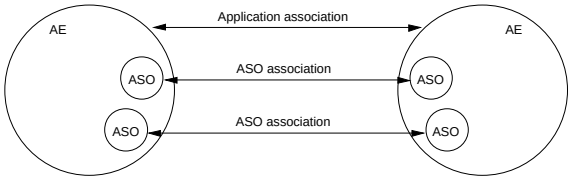


Figure 1-1. Application and ASO Associations

1.5.9 *ASO-context*: 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.

1.5.10 *Application-context*: a special type of ASO context that characterises the application association.

1.6 Rationale

1.6.1 The framework for the standardisation of the upper layers is based on the concept that a set of standardised common communication services be provided on which ATN user applications, or "Data Link Applications", can be constructed. These communication services can 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.

1.6.2 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.

1.6.3 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.

1.6.4 For CNS/ATM-1, only a simple communications service (the Dialogue Service - see 2) is defined. For the future it is planned to define further generic communication service profiles for use in the aeronautical domain, to provide applications with standard services to access the ATN. Each profile constitutes an upper layer "protocol stack" definition which when implemented provides the appropriate functionality in the selected upper layers.

1.6.5 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.

1.6.6 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, transfer syntaxes, etc. whenever a new DLA is introduced or an existing DLA modified.
e) The communication services may in some cases be based on COTS (Commercial Off The Shelf) products.
f) It does not require the implementation of application specific interactions (e.g. time-outs, message sequencing rules) within the communication service.

1.6.7 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.

1.6.8 ATN applications are defined to use the services of the selected ATN profiles. The communication aspects of these applications will then 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.)

1.6.9 Functions of the Upper Layers

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

1.6.11 It is therefore envisaged that a set of functions which add value to the ATN transport service will be standardised, to 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.

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

1.6.13 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.

1.6.14 "Fast Byte" Efficiency Enhancements

1.6.15 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. While it is possible to use the normal OSI layer protocol to signal that null functionality is 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, together with minimal functionality.

1.6.16 For example, if a transport layer fast byte were exchanged, the layer service remains the same, i.e., the transport fast byte is an option 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 would get a QoS

which is only as good as that provided by the underlying network service.

1.6.17 For the upper layers, the typical, full 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.

1.6.18 A null PCI optimisation for the data phase implies a reduction in the layer service available to the application. For instance, in the 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 optimisation.

1.6.19 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 (FUs). Orderly release of the session connection is conventionally 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 the no orderly release (NOR) 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 specifications that require the (non-negotiable) kernel to be indivisible.

1.7 Overview

1.7.1 The ATN upper layer architecture is based on the services provided by the ATN Internet, specifically the connection-mode transport service as defined in 5.5 of the Internet Communication Service SARPs.

1.7.2 The main aspects of the ATN Upper Layer Architecture are as follows:

a) ISO/IEC 9545, edition 2 specifies the extended application layer structure (ALS). This allows modular construction of protocols by specification of application service elements (ASEs). An ASE may be implemented as a software module. These are combined into Application Service Objects (ASOs). Interactions between ASEs and ASOs are mediated by a control function (CF).
b) ISO/IEC 8649, edition 2 and ISO/IEC 8650, edition 2 specify the Association Control Service Element (ACSE) needed to support the ALS. ACSE is required for the establishment and termination of application associations, using transport connections.
c) Amendments to ISO/IEC 8823 and ISO/IEC 8327 specify efficient Presentation Protocol and Session Protocol. The amendments specify protocol variants which are highly efficient in terms of the protocol overheads required, but which offer minimal functionality.

1.7.3 The scope of the ULCS SARPs and the relationship to the basic reference model for open systems

interconnection (RM-OSI, ISO/IEC 7498-1) are illustrated in Figure 1-2. This shows the scope of the upper layer architecture (ULA) as being the upper three layers of the OSI reference model. The application layer is further decomposed into Application Entities (AEs), which in turn are composed of Application Service Elements (ASEs). ASEs for ATN applications, as well as the "User" elements, are specified in the relevant parts of the ATN SARPs 2 and 3 SARPs.

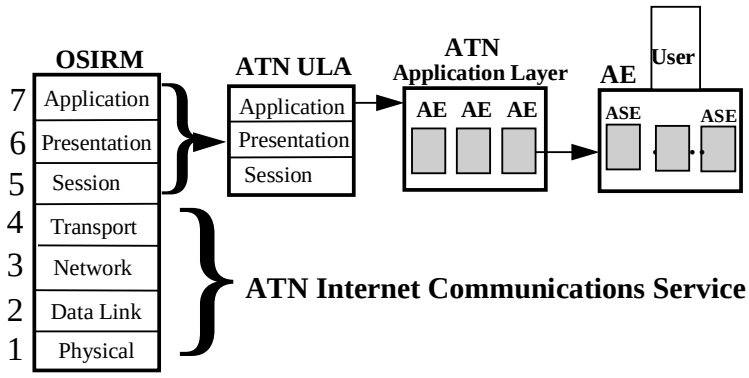


Figure 1-2. Conceptual view of the scope of the UL SARPs

1.7.4 With reference to figure 1-2, the effect of the session and presentation layer efficiency enhancements can readily be visualised:

a) a request to establish an association at the application layer is mapped to the connect request primitives of the supporting upper layers. Thus, A-ASSOCIATE maps to P-CONNECT which maps to S-CONNECT. The presentation and session layer connect protocols are compressed into a single octet each.

b) a data transfer request at the application layer is effectively mapped directly to the transport layer data transfer service, i.e. the presentation and session layer overheads in the data transfer phase are reduced to zero.

c) a request to release an established association at the application layer cannot be mapped to the disconnect request primitives of the supporting upper layers, because the ability to utilise session or presentation protocol control information has been sacrificed in order to obtain maximum efficiency in the data transfer phase. The only release mechanism available is an abrupt release of the underlying transport connection. An orderly release function is provided by the Control Function in the application layer using data requests (to carry the ACSE release protocol) and abort requests (to actually clear the connection after the orderly release.)

1.7.5 Description of the Application Layer

1.7.6 When application processes (APs) in different end systems need to co-operate 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.

For CNS/ATM-1, each application is embodied in a single AE.

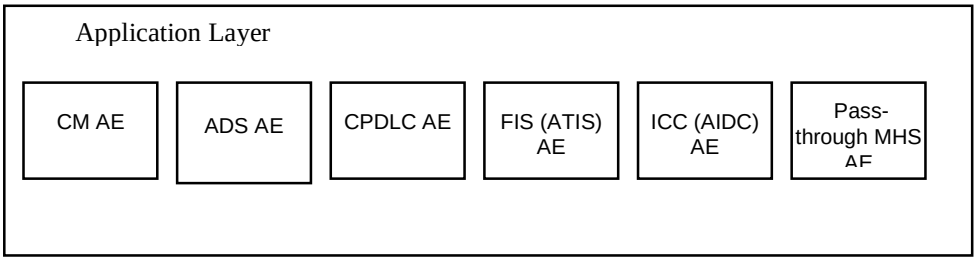


Figure 1-3. Conceptual view of Application Layer

1.7.7 Description of the AE Structure

1.7.8 Each AE contains an ATN ASE, which is the communication element responsible for an ATN application. In general, the internal structure of an AE may be of arbitrary complexity, but for CNS/ATM-1, the AE consists only of an ATN-App ASE (e.g. the ADS-air ASE), ACSE, and the Control Function (CF).

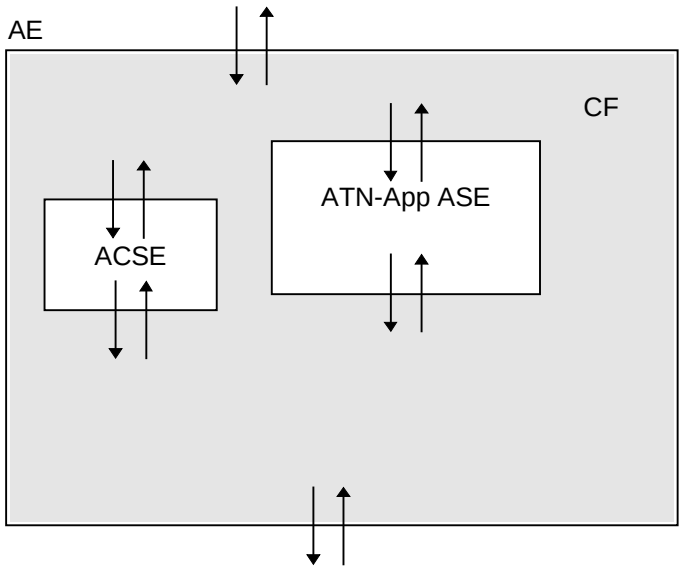


Figure 1-4. Specific CNS/ATM-1 Application Entity structure

1.7.9 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 and ACSE.

1.7.10 The internal structure of the ASE may be of arbitrary complexity and is not visible to the CF.

1.7.11 In CNS/ATM-1, there is no architectural capability for multiple instances of the same ATN-App 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 (and hence a single application association) over the lifetime of the ASE invocation.

1.7.12 There is no interaction inside an AE between applications of different types, since these are not provided by the CNS/ATM-1 architecture. Any requirement for interaction between applications occurs in

user space through, e.g., global data structures.

1.7.13 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.

1.7.14 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. The CF does not produce protocol data units; only ASEs and ASOs can do that.

1.7.15 In CNS/ATM-1, the upper AE service boundary is identical to the upper ATN application ASE service boundary, i.e., it is a transparent "pass-through" interface.

1.7.16 Support of Traditional OSI Applications

1.7.17 The ATN Upper Layer Communications Service (ULCS) offers support for all basic communications applications. Such basic communications applications are characterised in the OSI upper layers as requiring the support of only the kernel and full-duplex functional units of the session protocol. The ULCS offers a limited presentation transfer-syntax negotiation and session no-orderly-release statement. The ULCS thereafter (in the data transfer phase) offers a simple pass-through service. Thus, application protocols such as CMIP (in e.g., the AOM12 profile) are fully supported by the ULCS.

1.7.18 Generally, traditional applications not supported by the ULCS are those making use of session facilities, e.g., those using the Reliable Transfer Service Element (RTSE) for reliable bulk transfer. Such applications include Message Handling System (MHS) and certain uses of Directory.

1.7.19 Interoperability with Conventional OSI Stacks

1.7.20 The ISO upper layer efficiency enhancements were carefully designed to interoperate with conventional OSI stacks. First, the upper layer amendments are amendments to the base standards, not separate new protocol standards. The upper layer efficiency enhancements association establishment sequence is offered as the primary establishment choice, and if refused (or the transport connection is accepted without the return of the efficiency enhancement establishment sequence) or not interpreted by a conventional OSI upper layer stack, the transport connection is maintained and the traditional OSI establishment may be offered.

1.7.21 However, negotiation to full OSI is outside the scope of the ULCS SARPs. Thus an implementation of the ULCS will not interwork with a conventional OSI protocol stack unless special additional previsions are made. Thus it is possible to imagine a ground-based communications system which implements both the full and efficient modes of OSI and is able to negotiate at connection time so that it can work with either a ULCS implementation or a full OSI implementation.

1.8 Inter-relationships with Other SARPs

1.8.1 The ATN upper layer architecture is based on the services provided by the ATN Internet, specifically the connection-mode transport service as defined in the Internet Communication Service ATN SARPs 5, section 5.5.

1.8.2 The following CNS/ATM-1 application SARPs make use of the ULCS SARPs (ATN SARPs 4) to perform required dialogue service functions, and to define a profile of the ACSE, Presentation and Session protocol standards:

a) Automatic Dependent Surveillance (ADS) Application, ATN SARPs 2, Part 2.1
b) Automatic Dependent Surveillance Report Forwarding (ARF) Application, ATN SARPs 2, Part 2.2
c) Context Management (CM) Application, ATN SARPs 2, Part 1
d) Controller Pilot Data Link Communication (CPDLC) Application , ATN SARPs 2, Part 3
e) Flight Information Service (FIS) Application- Automatic Terminal Information Services (ATIS), ATN SARPs 2, Part 4.
f) ATS Message Handling Services (ATSMHS) - ATN Pass-through Service, ATN SARPs 3, Part 1.3.

1.8.3 The following CNS/ATM-1 application SARPs make use of the ULCS SARPs (ATN SARPs 4) only to define a profile of the ACSE, Presentation and Session protocol standards:

a) ATS Interfacility Data Communications (AIDC) Application, ATN SARPs 3, Part 2

1.9 Structure of ULCS SARPs

1.9.1 The ULCS SARPs constitute ATN SARPs 4 of the CNS/ATM-1 SARPs, and have the following structure:

a) Introduction (4.1) contains the purpose and structure of the UL Communications Service Specification, and a background to the functionality defined herein.

b) Dialogue Service Description (4.2) describes the abstract service which is defined for application specifications to refer to in order to provide a common communications service.

c) Application Entity (AE) Description (4.3) describes the Application Entity and specifies the Control Function (CF) which co-ordinates the operation of the various Application Service Elements (ASEs). It also describes the names which are assigned to various upper layer entities.

d) Session Layer Requirements (4.4) describes the requirements for the OSI Session Layer, in the form of a Profile Requirements List (PRL).

e) Presentation Layer Requirements (4.5) describes the requirements for the OSI Presentation Layer, in the form of a PRL.

f) ACSE Specification (4.6) describes the requirements for the Association Control Service Element (ACSE).

1.10 References

1.10.1 Automatic Dependent Surveillance Application, Annex 10, Volume III, Part 1, Chapter 3 (ATN), Appendix A, ATN SARPs II - Air-Ground Applications, section 2.2.1

1.10.2 Automatic Dependent Surveillance Report Forwarding Application, Annex 10, Volume III, Part 1, Chapter 3 (ATN), Appendix A, ATN SARPs II - Air-Ground Applications, section 2.2.2

1.10.3 Context Management Application, Annex 10, Volume III, Part 1, Chapter 3 (ATN), Appendix A, ATN SARPs II - Air-Ground Applications, section 2.1

1.10.4 Upper Layer Communications Service, Annex 10, Volume III, Part 1, Chapter 3 (ATN), Appendix A, ATN SARPs IV

2. DIALOGUE SERVICE

2.1 Introduction

2.1.1 The dialogue service is a service that allows a user to bind to an association, send data, and unbind from the association. It is the lower service boundary used by ATN-App ASEs, i.e. it is the ASE's "world view".

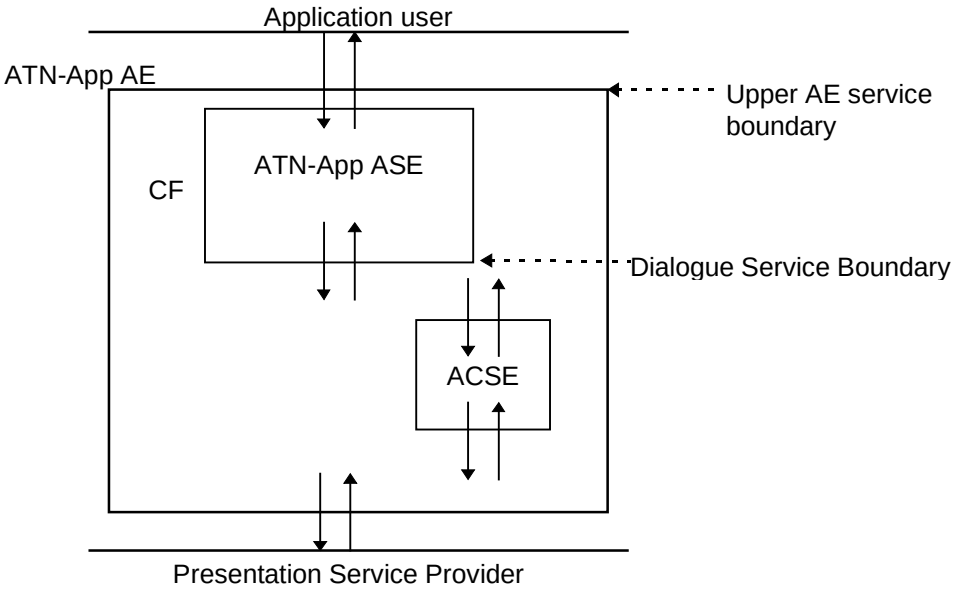


Figure 2-1. Position of Dialogue Service Boundary

2.1.2 Connection Mode

2.1.3 The dialogue service provides a connection-oriented 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.

2.2 Description and Rationale

2.2.1 Each OSI application contains an element, the application entity (AE), which provides the services for communicating with a peer application. AEs use the services provided by ASEs, including those provided by certain, standard ASEs such as ACSE. ASEs themselves provide a defined set of services using a subset of OSI upper layer services.

2.2.2 On analysis of the requirements of CNS/ATM-1 Package applications, the services required by the ATN applications from the ATN upper layers were found to be fairly simple (apart from the efficiency enhancements required to the upper layer protocols themselves). The Dialogue Service (DS) is a technique which employs a formal, or abstract syntax to be used in specifying how an ATN ASE uses the ATN upper layer services. Conceptually, the Dialogue Service 'shields' the user - the ASE - from the complexity of the upper layers, and provides the common functionality required by the set of ASEs. The use of the word 'abstract' to describe the DS means that the elements of the syntax - the Dialogue Service primitives - need

not be implemented as described, or even at all. What is required is that, when an ASE has been specified using the DS, an implementation of that ASE behaves exactly as defined in the SARPs for the DS syntax specified.

2.2.3 The DS has three main attributes:

a) It specifies a standard and sufficient set of communication primitives for use by ASEs;
b) It enables a CF to be defined to map these primitives to ATN upper layer protocol elements in a consistent way;
c) It **does not,** and can not, specify the mapping of any users' API, either to the DS or directly to ATN upper layer protocol elements - this is up to the implementor.

2.2.4 With regard to the last of these points, the ATN application SARPs for CNS/ATM-1 define the communication services required by each AP in terms of an abstract service, and specify the mappings between this service and the DS. Again, the service is described as abstract because it is simply a technique used to describe the requirements, and the actual implementation of the service will be a local matter i.e., will depend on the API used.

2.2.5 The developer therefore has two choices. As several ATN ASEs have been defined using the syntax of the DS, one choice would be to observe the mapping between the DS primitives and ATN upper layer protocol elements as given in the SARPs, and invoke the corresponding protocols directly from the API (and perform the reverse mapping for 'inbound' UL events). The second choice would be to develop a module which provided the required mapping between UL protocol elements and some, local representation of the DS, and then invoke the primitives of this DS directly from each API. The latter approach has several advantages:

a) The mapping of an API directly to ATN UL protocols is a fairly complex and error-prone process, not to be undertaken lightly, especially when such a mapping already exists, albeit for an abstract syntax;
b) Inter-operability testing is facilitated by identical mappings at both ends of a connection - this is much easier to achieve when the DS is used;
c) When three or more ASEs have to be developed, use of the DS will require less programming effort

2.3 Scope of the Dialogue Service

2.3.1 The DS allows two users (ASEs) to establish a dialogue (an application association), exchange data and terminate the dialogue in an orderly manner, with no data lost. The DS also allows a user to terminate the dialogue abruptly - with potential loss of data - and to be informed when an error in the underlying ATN service causes the dialogue to be abruptly terminated by the service provider.

2.3.2 Note that, by including user data in the D-START primitives, if the D-START response is sent with a result code of 'Rejected', with or without data, the dialogue (association) is terminated automatically (in fact the association is never actually set up). This technique provides what is, in effect, a datagram service via the DS.

2.3.3 The DS primitives and descriptions listed in the SARPs are repeated below for convenience.

Table 2-1. Summary of Dialogue Service primitives

Service	Description	Parameters
D-START	This is a confirmed service used to establish the binding between the communicating DS-Users.	*Called Peer ID* *Calling Peer ID* *DS-User Version Number* *Security Requirements* *Quality-of-Service* *Result* *Reject Source* *User Data*
D-DATA	This unconfirmed service is used by a DS-User to send a message from that DS-User to the peer DS-User.	*User Data*
D-END	This is a confirmed service used to provide the orderly unbinding between the communicating DS-Users, such that any data in transit between the partners is delivered before the unbinding takes effect.	*Result* *User Data*
D-ABORT	This unconfirmed service can be invoked to abort the relationship between the communicating DS-Users. Any data in transit between them may be lost.	*Originator* *User Data*
D-P-ABORT	This unconfirmed service is used to indicate to the DS-User that the dialogue service provider has aborted the relationship with the peer DS-User. Any data in transit between the communicating DS-Users may be lost.	*(no parameters)*

2.4 Mapping of Dialogue Service Primitives

2.4.1 Figures 2-2 and 2-3 illustrate the sequence of OSI services that are invoked following the issuance of a D-START request. Two cases arise since the transport service connection request is limited to 32 octets of user data, which must include the protocol control information (PCI) resulting from the protocols supporting the A-ASSOCIATE, P-CONNECT and S-CONNECT services.

2.4.2 In the first case, as shown in Figure 2-2, the amount of user data included with the D-START is little enough that everything fits into the 32 octets of the T-CONNECT request user data.

2.4.3 In the second case, shown in Figure 2-3, the amount of user data sent with the D-START is too much to fit into the T-CONNECT request; accordingly the PCI and the D-START user data are sent in a T-DATA PDU which is sent after the transport connection has been established. Note that this requires an extra round-trip over the data link, which in some cases could be a significant overhead.

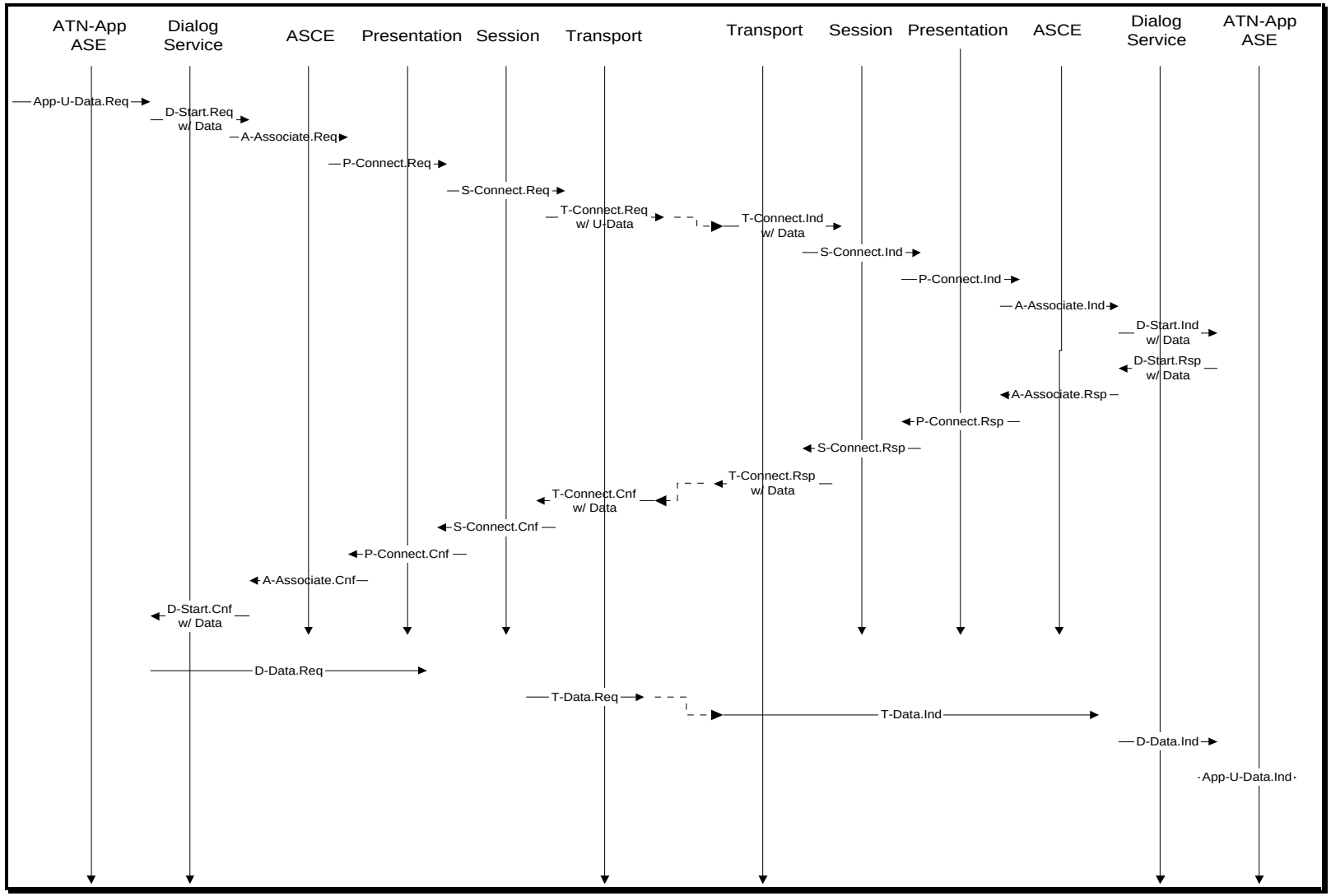

ATN-App ASE
Dialog Service
ASCE
Presentation
Session
Transport
Transport
Session
Presentation
ASCE
Dialog Service
ATN-App ASE
App-U-Data.Req
D-Start.Req w/ Data
A-Associate.Req
P-Connect.Req
S-Connect.Req
T-Connect.Req w/ U-Data
T-Connect.Ind w/ Data
S-Connect.Ind
P-Connect.Ind
A-Associate.Ind
D-Start.Ind w/ Data
D-Start.Rsp w/ Data
A-Associate.Rsp
P-Connect.Rsp
S-Connect.Rsp
T-Connect.Rsp w/ Data
T-Connect.Cnf w/ Data
S-Connect.Cnf
P-Connect.Cnf
A-Associate.Cnf
D-Start.Cnf w/ Data
D-Data.Req
T-Data.Req
T-Data.Ind
D-Data.Ind
App-U-Data.Ind

Figure 2-2.

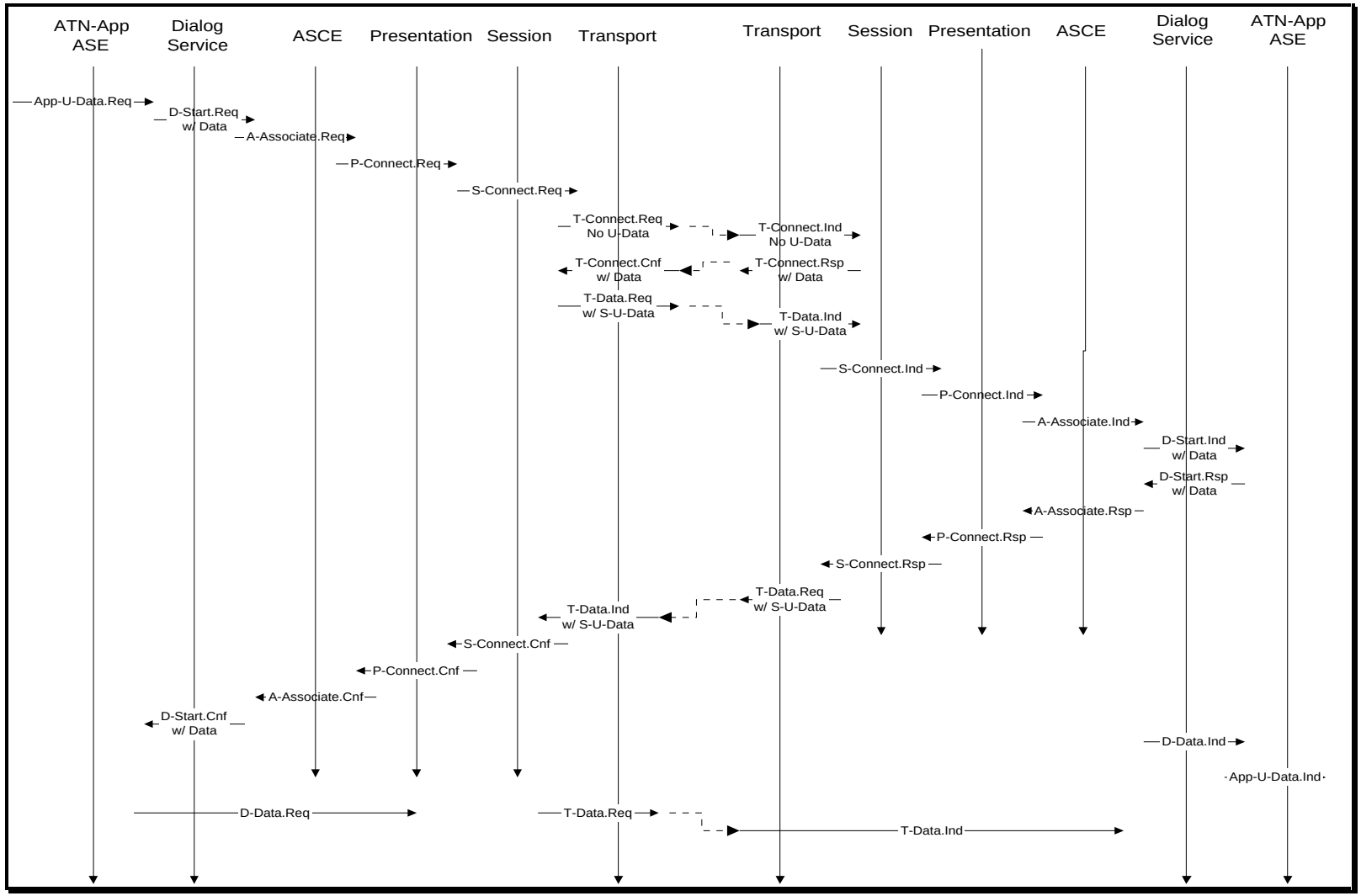

ATN-App ASE
Dialog Service
ASCE
Presentation
Session
Transport
Transport
Session
Presentation
ASCE
Dialog Service
ATN-App ASE
App-U-Data.Req
D-Start.Req w/ Data
A-Associate.Req
P-Connect.Req
S-Connect.Req
T-Connect.Req No U-Data
T-Connect.Ind No U-Data
T-Connect.Cnf w/ Data
T-Connect.Rsp w/ Data
T-Data.Req w/ S-U-Data
T-Data.Ind w/ S-U-Data
S-Connect.Ind
P-Connect.Ind
A-Associate.Ind
D-Start.Ind w/ Data
D-Start.Rsp w/ Data
A-Associate.Rsp
P-Connect.Rsp
S-Connect.Rsp
T-Data.Req w/ S-U-Data
T-Data.Ind w/ S-U-Data
S-Connect.Cnf
P-Connect.Cnf
A-Associate.Cnf
D-Start.Cnf w/ Data
D-Data.Ind
App-U-Data.Ind
D-Data.Req
T-Data.Req
T-Data.Ind

Figure 2-3.

2.5 QoS parameters

2.5.1 Some applications may be invoked as either ATSC or AOC applications, e.g. ADS or CPDLC. There is a constraint in ADS to monitor the number of ADS applications running on board in order to allow the establishment of ADS contracts on the aircraft by four ATCCs. As a consequence, it is necessary to be able in the aircraft to discriminate ATSC ADS applications and AOC ADS applications.

2.5.2 There are two potential ways to do that:

1. Use different ATN addresses for AOC and ATSC. Locally it would be possible to recognise the type of the calling ADS. This is not possible within the current naming scheme, which identifies only one type for each application.
2. Get the traffic type indication from the ATN Internet and from the upper layers. The traffic type is conveyed in the QoS parameter of the D-START primitives. It is embedded in the Routing Class parameter.

2.5.3 The Routing Class parameter contains the traffic type indication, as shown in the application SARPs in the sections headed "Dialogue Service Requirements". Thus, an ASE receiving a D-START indication knows if the peer ASE is establishing an ATSC or an AOC dialogue.

2.5.4 The ULCS are not specific to ATSC applications. The default traffic type is "General Communication", except for the air-ground applications, which depend on the default value of "ATSC: No Traffic Type Policy Preference" (e.g. see CM 2.1.7.1, ADS 2.2.1.7.1.2, CPDLC 2.3.7.1.1.2, FIS 2.4.7.2.1). However, any user of the Dialogue Service can override this default and set to traffic type to ATSC, AOC, AAC or SMC by selecting an appropriate value of Routing Class.

2.5.5 An AOC ASE will provide the ULCS with a routing class related to AOC routes whereas an ATSC ASE will provide routing class related to ATSC applications.

2.5.6 If a user process needs to know what is the nature of the communication (ATSC or AOC), the application interface includes the traffic type as parameter, in addition to the Class Of Communication. Thus the ADS-air-user will be able to monitor the number of ATSC and AOC links and take the appropriate action when the number of ATSC links becomes too low.

2.5.7 For the “General Communications” applications, the ULCS will not generate any security label. Thus, non-ATN intermediate systems can be used for this traffic. No security parameters are encoded for General Communications.

2.6 D-ABORT Service Peculiarities

2.6.1 If a user of the Dialogue Service invokes a D-ABORT request before the dialogue has been fully established, then the peer user can receive a D-P-ABORT (provider abort) indication, rather than the D-ABORT indication which might have been expected. Any user data on the D-ABORT request will be lost. The reason for this is as follows. The ULCS uses a null encoding / short connect scenario.

a) When application App A issues an D-START request and hence an A-ASSOCIATE request, ACSE A goes into the Awaiting AARE state.
b) If App A decides not to wait for the A-ASSOCIATE confirmation, and issues A-ABORT request,

then ACSE A sends an ABRT APDU as User-data on a P-U-ABORT request primitive.

c) As the CF is not yet in the DATA TRANSFER state, it must map the P-U-ABORT request straight through to the supporting Presentation service.
d) PPM A then issues an S-U-ABORT request with no SS-user-data (In fact the PPM attempts to follow clause 6.4.4.1 of the presentation protocol standard and send an ARU PPDU, but this is not conveyed over the presentation-connection, as null encoding is selected - 5.4bis.4 of 8823-1 AM1). The text in 8823-1 AM1 is not very clear in this area.
e) SPM A then issues a T-DISCONNECT request with the first octet of user data set to 01. (7.84.1 of 8827-1 AM1).
f) In the peer system, SPM B therefore receives a T-DISCONNECT indication with the first octet of user data set to 01, and therefore issues an S-U-ABORT indication to its user.
g) PPM B receives this S-U-ABORT indication with no SS-user-data, interprets this as an ARP PPDU, and issues a P-P-ABORT indication (6.4.4.6 of 8823-1 AM2).
h) ACSE B receives the P-P-ABORT indication and issues an A-P-ABORT indication. The CF maps this to a D-P-ABORT indication which is passed to App B.

2.6.2 Thus, what started off as an ACSE-user abort at A is reported as an ACSE-provider abort at B. Any user-data that was in the A-ABORT request is of course lost.

2.6.3 This seemingly strange behaviour arises because the receiving PPM (B in this case) cannot distinguish whether the abort originated in the peer PPM or the peer PS-user. In a null-encoding regime, the presentation entities cannot communicate any PCI to resolve this. (The session layer cheats by adding one octet to the T-DISCONNECT user data).

2.6.4 It could be argued that the PS-provider is incapable of honouring the user abort request, so it does the next best thing and aborts the association on behalf of the user.

2.6.5 The application SARPs cannot depend on a D-ABORT request issued during the connection phase necessarily causing a D-ABORT indication at the peer. (If the application waits for the end of the connection phase, then all is well, as the ULCS SARPs will map the ABRT + any user data onto a P-DATA request).

2.6.6 The abort mechanism should be seen as a "crash close" and is not suitable for the user to convey any subtle semantics depending upon the originator of the abort.

3. APPLICATION ENTITY

3.1 Naming, addressing and registration

Note.— This section has been moved to Annex A. It is proposed to remove this material from this document altogether and include it instead in Part II of the Comprehensive ATN Manual (CAMAL).

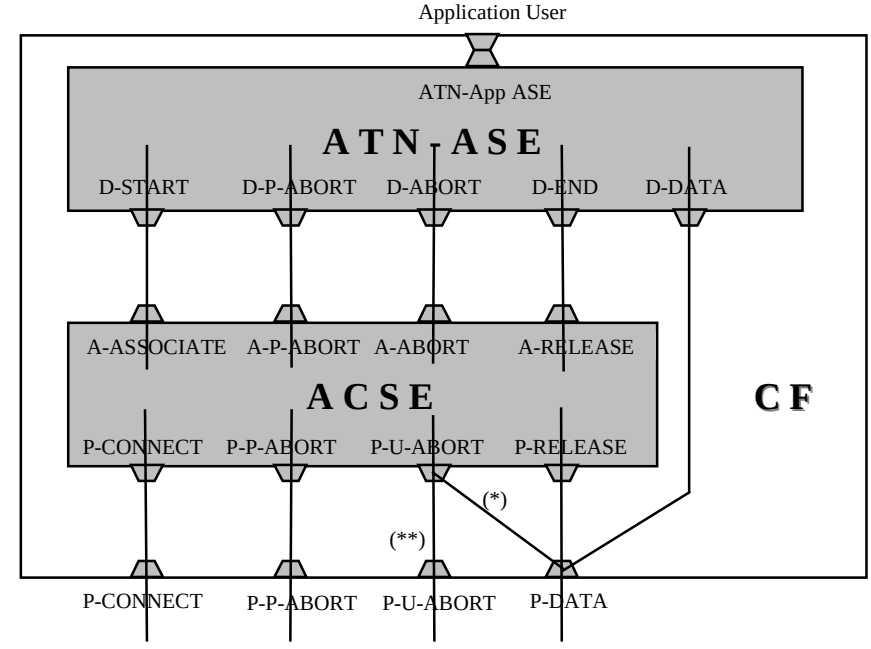
3.1.1 Guidance material on upper layer naming and addressing is given in Part II of the Comprehensive ATN Manual.

3.2 Control Function

3.2.1 The Control Function (CF) controls the interactions between the ASEs and governs the behaviour of the AE as seen at the upper and lower service boundaries.

3.2.2 The CF interacts with the constituent ASEs and ASOs by means of the abstract services defined at their upper and lower service boundaries. Thus, the CF "intercepts" the Presentation service primitives invoked by ACSE at its lower service boundary, and re-maps them to provide the required CNS/ATM-1 upper layer functionality.

3.2.3 The air-ground CF is described in the following figure. The basic five threads of CF functionality are as described in the following paragraphs.



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

Figure 3-4. Primitive Mappings by the Control Function

3.2.4 The D-START service is mapped to the A-ASSOCIATE service of ACSE, which in turn is mapped by ACSE to the P-CONNECT service of the presentation layer. The P-CONNECT service is then mapped to the S-CONNECT service of the session layer, which in turn maps to the T-CONNECT service, or, if a

suitable transport connection already exists, to the T-DATA service.

3.2.5 When the DS-User invokes a D-DATA request primitive, this is mapped by the CF to P-DATA, which in CNS/ATM-1 is equivalent to the T-DATA service offered by the ATN Internet.

3.2.6 When the DS-User invokes a D-END request primitive, this is mapped by the CF to the A-RELEASE service of ACSE, which results in a P-RELEASE request at the ACSE lower service boundary. In order to ensure that no data is lost without notification (the "orderly release" function), the P-RELEASE is re-mapped by the CF to the P-DATA service. The CF terminates the connection once the D-END service has completed.

3.2.7 When the DS-User invokes a D-ABORT request primitive, this is mapped by the CF to the A-ABORT service of ACSE. The A-ABORT is mapped by ACSE to the P-U-ABORT service. A P-U-ABORT with user-data is re-mapped by the CF to P-DATA which is followed by a P-U-ABORT request. A P-U-ABORT without user-data is mapped directly to the P-U-ABORT service.

3.2.8 The D-P-ABORT occurs as an indication (‘up’) only. The P-P-ABORT service is mapped to the A-P-ABORT service. Alternatively, and internal ACSE error can result in A-P-ABORT being generated. The A-P-ABORT service is mapped by the CF to the D-P-ABORT service.

3.2.9 CF Mapping Flow Diagrams

3.2.10 The role of the CNS/ATM-1 Control Function (CF) is to moderate the interaction of the ATN ASE and the ACSE. To that end, descriptions of the characteristic event sequences are provided in the figures below.

3.2.11 The first diagram shows the sequence of events when a DS-User issues a D-START request primitive. This is mapped by the CF to an A-ASSOCIATE request, and the CF enters the ASSOCIATION PENDING state (STA 1) as the Initiator of the dialogue. ACSE processes the A-ASSOCIATE primitive, and if all is well will issue a P-CONNECT request to establish a presentation connection to support the association. Subsequently, a P-CONNECT confirmation is expected from the presentation service. This is delivered by the CF to ACSE, which interprets the embedded AARE APDU and generates an A-ASSOCIATE confirmation primitive, which may be either positive or negative (determined by the value of the Result parameter). If positive, the CF issues a D-START confirmation to the DS-User and enters the DATA TRANSFER state (STA 2). If negative, the CF issues a D-START confirmation to the DS-User and returns to the NULL state (STA 0), effectively ceasing to exist for this invocation.

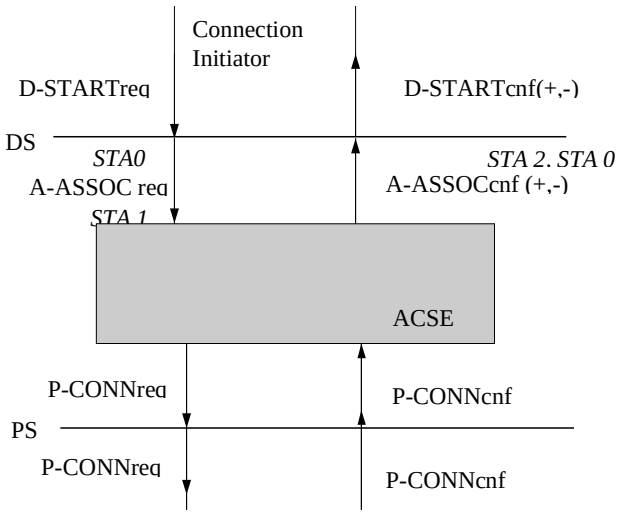


Figure 3-5.

3.2.12 The next diagram shows the same sequence of events from the viewpoint of the communication peer. When a P-CONNECT indication is received from presentation service, the CF passes this to ACSE and enters the ASSOCIATION PENDING state (STA 1) as the Responder of the dialogue. ACSE interprets the embedded AARQ APDU and generates an A-ASSOCIATE indication primitive, which is mapped by the CF into a D-START indication and delivered to the DS-User. Subsequently, a D-START response is expected from the DS-User, which may be either positive or negative (determined by the value of the Result parameter). This is mapped by the CF to an A-ASSOCIATE response. ACSE processes the A-ASSOCIATE primitive, and if all is well will issue a P-CONNECT response. This is delivered by the CF to the presentation service. If the D-START response was positive, the CF enters the DATA TRANSFER state (STA 2), and the dialogue is established. If negative, the CF returns to the NULL state (STA 0), effectively ceasing to exist for this invocation.

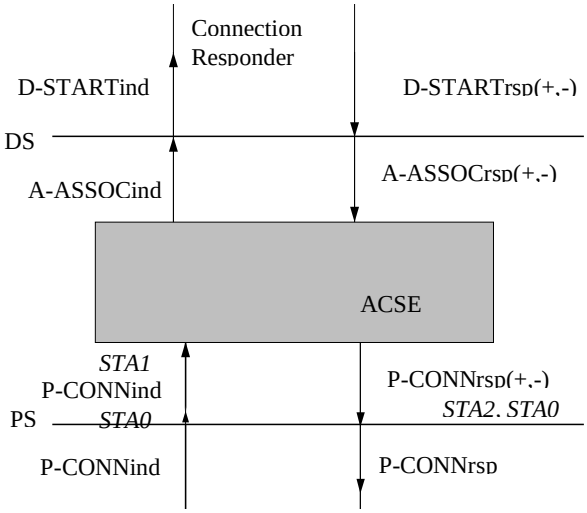


Figure 3-6.

3.2.13 The following diagram shows transfer of user data after a dialogue has been established and the

peer CFs are in the DATA TRANSFER state (STA 2). When a P-DATA indication is received from presentation service, the CF passes the user data to the ATN application ASE. The subsequent action of the ASE is application-specific, and may include invoking end-user primitives at the upper AE service boundary. Conversely, when the DS-user (i.e. the ATN application ASE) wishes to send data to the remote peer, (for example after receiving a stimulus from the end-user at the upper AE service boundary,) it invokes a D-DATA request. This is wrapped with appropriate headers by the CF and mapped onto a P-DATA request primitive.

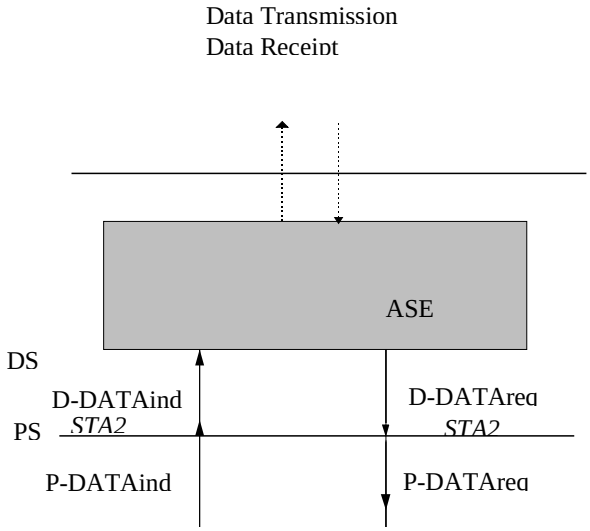


Figure 3-7.

3.2.14 The next diagram illustrates the normal release sequence from the point of view of the release initiator. The DS-User issues a D-END request primitive. This is mapped by the CF to an A-RELEASE request, and the CF enters the RELEASE PENDING state (STA 3) as the Release Initiator. ACSE processes the A-RELEASE primitive, and if all is well will issue a P-RELEASE request to terminate the presentation connection underlying the association. In order not to disrupt any data which is in transit, (recalling that the Session Orderly Release function is not being used), the CF re-maps the P-RELEASE to a P-DATA request , which will contain an ACSE RLRQ APDU as user information. Subsequently, a P-DATA indication containing an ACSE RLRE APDU is expected from the presentation service. The RLRE may be positive or negative, depending upon whether the peer user agreed to the termination of the dialogue. This is re-mapped by the CF into a P-RELEASE confirmation and delivered by the CF to ACSE, which interprets the embedded RLRE APDU and generates an A-RELEASE confirmation primitive, which again may be either positive or negative (determined by the value of the Result parameter). If negative, the CF issues a negative D-END confirmation to the DS-User and returns to the DATA TRANSFER state (STA 2), just as if the D-END request had never been issued. If positive, the CF issues a positive D-END confirmation to the DS-User, issues a P-U-ABORT request to really terminate the presentation connection, and enters the NULL state (STA 0), effectively ceasing to exist for this invocation.

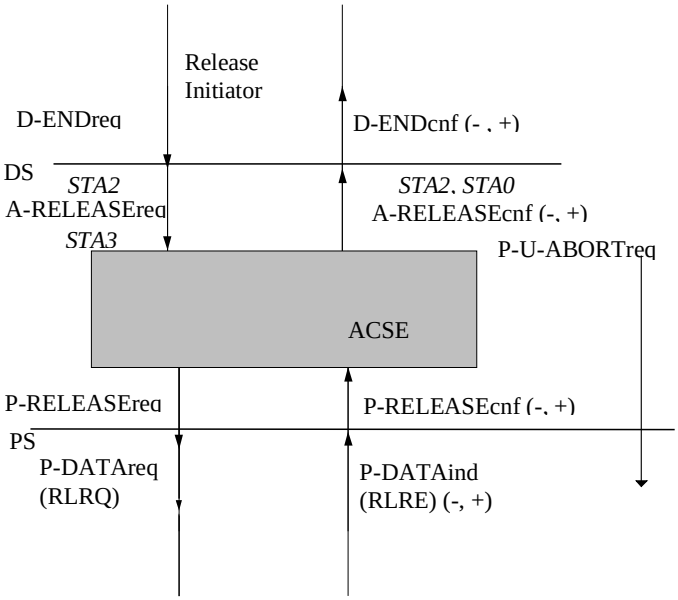


Figure 3-8.

3.2.15 The next diagram illustrates the normal release sequence from the point of view of the release responder. When a P-DATA indication containing a RLRQ APDU as user information is received from presentation service, the CF re-maps this to a P-RELEASE indication, passes this to ACSE and enters the RELEASE PENDING state (STA 3) as the Release Responder. ACSE processes the P-RELEASE primitive, and if all is well will issue an A-RELEASE indication at its upper service boundary. This is passed by the CF to the DS-User as a D-END indication. Subsequently, a D-END response is expected from the DS-User, which may be either positive or negative (determined by the value of the Result parameter). This is mapped by the CF to an A-RELEASE response. ACSE processes the A-RELEASE primitive, and if all is well will issue a P-RELEASE response containing a RLRE APDU as user information. The RLRE may be positive or negative, depending upon whether the DS-User agreed to the termination of the dialogue. This is re-mapped by the CF to P-DATA and delivered the presentation service. If the D-END response was negative, the CF returns to the DATA TRANSFER state (STA 2)), just as if the D-END indication had never been issued. If positive, the CF enters the NULL state (STA 0), effectively ceasing to exist for this invocation.

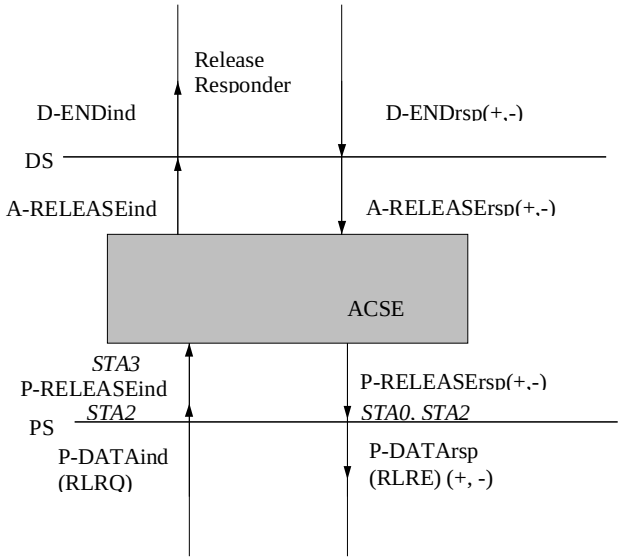


Figure 3-9.

3.2.16 The final pair of diagrams in this sequence illustrates the more complex case of "release collision". This can occur when both users of a dialogue issue D-END requests near-simultaneously. In such a case, the behaviour of the CF depends upon whether it was the Initiator of the dialogue or the Responder.

3.2.17 Firstly, the release collision is described from the point of view of the dialogue Initiator. The DS-User issues a D-END request primitive. This is mapped by the CF to an A-RELEASE request, and the CF enters the RELEASE PENDING state (STA 3) as the Release Initiator. ACSE processes the A-RELEASE primitive, and if all is well will issue a P-RELEASE request to terminate the presentation connection underlying the association. In order not to disrupt any data which is in transit, (recalling that the Session Orderly Release function is not being used), the CF re-maps the P-RELEASE to a P-DATA request , which will contain an ACSE RLRQ APDU as user information. Subsequently, a P-DATA indication containing an ACSE RLRE APDU is expected from the presentation service.

3.2.18 Instead of this, however, a P-DATA indication containing a RLRQ may be received, indicating that a release collision has occurred. The CF enters the RELEASE COLLISION state (STA 4) and delivers a P-RELEASE indication to ACSE, which interprets the embedded RLRQ APDU and generates an A-RELEASE indication primitive. The CF forms a D-END confirmation but does not yet deliver it to the DS-User. Instead, it issues A-RELEASE response to ACSE. ACSE will then issue a P-RELEASE response at its lower service boundary, containing a RLRE APDU as user information. This is re-mapped by the CF to a P-DATA request. Subsequently, a P-DATA indication containing an ACSE RLRE APDU is expected from the presentation service. This is re-mapped by the CF into a P-RELEASE confirmation and delivered by the CF to ACSE, which interprets the embedded RLRE APDU and generates a positive A-RELEASE confirmation primitive. The CF then issues the previously-formed D-END confirmation to the DS-User, issues a P-U-ABORT request to really terminate the presentation connection, and enters the NULL state (STA 0), effectively ceasing to exist for this invocation.

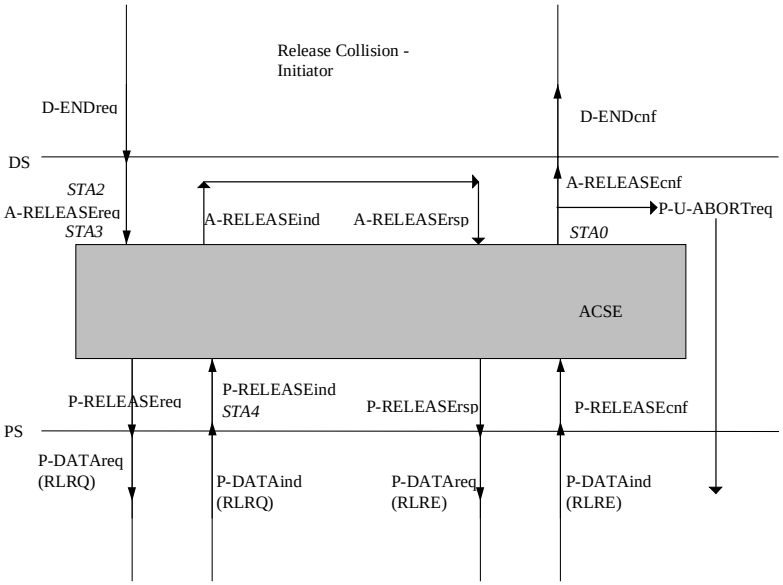


Figure 3-10.

3.2.19 Finally, the release collision is described from the point of view of the dialogue Responder. The DS-User issues a D-END request primitive. This is mapped by the CF to an A-RELEASE request, and the CF enters the RELEASE PENDING state (STA 3) as the Release Initiator. ACSE processes the A-RELEASE primitive, and if all is well will issue a P-RELEASE request to terminate the presentation connection underlying the association. As before, the CF re-maps the P-RELEASE to a P-DATA request, which will contain an ACSE RLRQ APDU as user information. Subsequently, a P-DATA indication containing an ACSE RLRE APDU is expected from the presentation service.

3.2.20 Instead of this, however, a P-DATA indication containing a RLRQ may be received, indicating that a release collision has occurred. The CF enters the RELEASE COLLISION state (STA 4) and delivers a P-RELEASE indication to ACSE, which interprets the embedded RLRQ APDU and generates an A-RELEASE indication primitive. The CF forms a D-END confirmation but does not yet deliver it to the DS-User. Instead, it waits for the peer to respond to the RLRQ previously sent. Subsequently, a P-DATA indication containing an ACSE RLRE APDU is received from the presentation service. This is re-mapped by the CF into a P-RELEASE confirmation and delivered by the CF to ACSE, which interprets the embedded RLRE APDU and generates a positive A-RELEASE confirmation primitive. The CF then issues the previously-formed D-END confirmation to the DS-User, and issues an A-RELEASE response to ACSE. This results in a P-RELEASE response being invoked by ACSE. The CF re-maps this to a P-DATA request and enters the NULL state (STA 0), effectively ceasing to exist for this invocation. The presentation connection will be terminated by the peer issuing P-U-ABORT request.

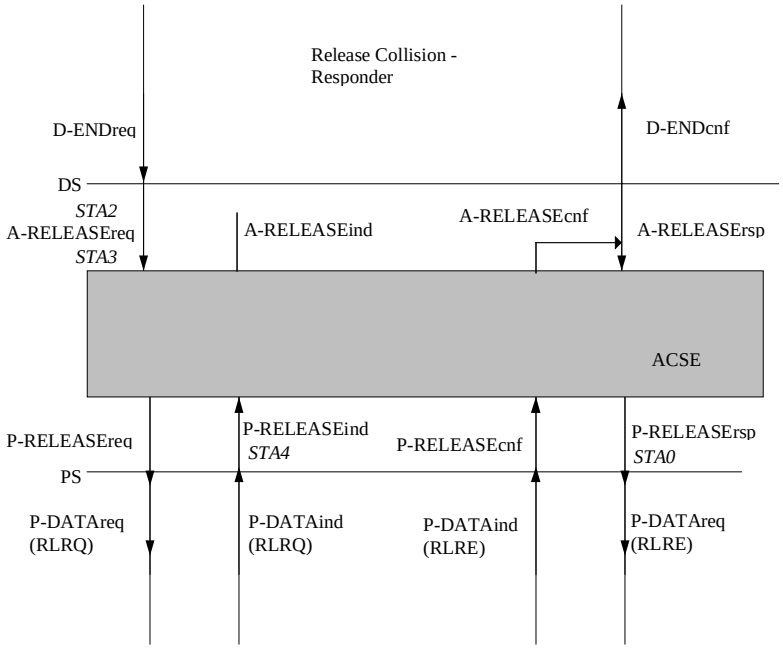


Figure 3-11.

3.3 Orderly Release

3.3.1 As Orderly Release is negotiated out of the Session layer, the possibility arises that data in transit could be lost without notification during the normal release of an association. This risk is eliminated by including provisions for Orderly Release in the CF specification. These provisions entail re-mapping the P-RELEASE primitives invoked by ACSE into P-DATA primitives (see 3.2.14 and 3.2.15 in this Guidance Material).

3.3.2 Only after the ACSE release APDUs have been safely exchanged, is the underlying connection released by the CF using the P-U-ABORT service (which maps ultimately to T-DISCONNECT). It is the responsibility of the release initiator to issue the P-U-ABORT request. If the release initiator does not behave as expected (e.g. due to faulty implementation, or internal failure), then it is possible that the Presentation Connection could be left intact after the association is released. This could cause depletion of resources.

3.3.3 The specification assumes that CFs behave according to the model, and the failure scenario described above is considered to be a local (implementation) issue. However, a Recommendation is made in ULCS SARPs 4.3.3.5.5.2.5 that the release responder should start a timer after entering the NULL state, and should itself release the underlying connection if the release initiator does not behave as expected within a certain time period.

3.4 Invalid State / Event Combinations

3.4.1 The CF is specified by means of a State Table, which is then described in plain text (if any conflicts arise, the text descriptions take precedence over the State Table).

3.4.2 “Blank cell” conditions in the State Table are supposed to be “impossible”. Therefore, if a valid

scenario is found such that a blank cell in the ULCS state table is encountered, then there is a defect in the ULCS SARPs.

3.5 When is it valid to invoke primitives?

3.5.1 Application user requests and responses may be invoked when the application ASE is in an appropriate state to receive them. Note that the CF model assumed for specification purposes is not allowed to queue user requests and responses until the ASE is ready to receive them, since all the processing must be completed in a single thread (ref. 3.3.1.2.4). The CF has no knowledge or visibility of the internal state of the ATN-App ASE. Thus, an implementation must handle the case where application user primitive invocations are rejected by the ATN-App ASE.

3.6 ACSE-detected errors

3.6.1 If the ACSE protocol machine detects a protocol error (unexpected APDU received, or invalid field encountered during processing of incoming APDU), then according to ISO/IEC 8650, it

a) issues an A-ABORT indication to its service-user, and
b) subsequently issues an ABRT APDU as user data on a P-U-ABORT request primitive.

3.6.2 The A-ABORT indication causes the CF to move to the NULL state, thus the P-U-ABORT request from the ACSE lower service boundary must be accepted by the CF when in the NULL state. The P-U-ABORT will always have an ABRT APDU as User-data, and the abort source field will be “ACSE service-provider” (see 7.3.3.4 of the ACSE protocol standard).

4. SESSION

4.1 Session Layer Functionality

4.1.1 The full OSI session protocol offers a rich selection of functional units with corresponding protocol mechanisms to support them. For basic communication applications, most of the functionality is not required, but the residual protocol overheads may still be excessive for bandwidth-limited communication paths.

4.1.2 The efficiency amendment to the session service standard specifies the no orderly release (NOR) 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 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.

4.1.3 The ATN upper layers use the Short Connect and Null Encoding protocol mechanisms to achieve a session protocol with minimal overheads. In order to achieve this, the only Session functional units selected for ATN are:

a) Kernel
b) No Orderly Release (NOR)

4.1.4 NOR is a "negative" function, which removes the ability to perform the orderly release of a session connection from the Session kernel. This means that data may be lost during the release of a connection without either user being informed. To overcome this, an orderly release function is provided by the control function defined in the ATN ULCS SARPs.

4.1.5 The efficiency amendment to the session protocol standard defines a number of protocol options, namely:

a) null-encoding protocol option
b) short-connect protocol option
c) short-encoding protocol option

4.1.6 The first two of these are selected for the CNS/ATM-1 profile, and are summarised briefly below.

4.1.7 *Null-encoding protocol option.* This is an option of the session protocol, negotiated during connection establishment, that permits a data transfer phase with zero session protocol control information (PCI) and without the ability to signal the orderly release of the session connection.

4.1.8 *Short-connect protocol option.* The negotiation of the null encoding protocol option can be done using the protocol options field of the conventional session establishment SPDUs. However, there is also the possibility of using the short-connect 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.

4.1.9 It is expected that the short-connect 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.

4.1.10 This is achieved as follows: the Short Connect SPDU — which is the short-encoding version of the full 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.

4.1.11 Previous 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 full protocol. 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.

4.1.12 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 achievable as there is no guarantee that the responding session entity, if based on the full protocol standards, will send a REFUSE SPDU to signal a protocol error, which is what a short-encoding for an SPDU would be.

4.2 Short SPDU Use and Encoding

4.2.1 The efficiency enhancements to the session protocol include the definition of "short" session protocol data units (SPDUs) which are distinguishable from the conventional longer form SPDUs.

4.2.2 The short-form SPDUs contain the following fields:

a) an SPDU identifier and parameter indication (SI&P) field of one octet
b) zero, one or more parameter fields, specific to the SPDU type
c) either one unspecified-length parameter, if defined for the SPDU type, or the optional User-information field.

4.2.3 For the simple session profile defined for CNS/ATM-1, the only fields present are the SI&P octet and the User-information field.

4.2.4 The structure of the SI&P octet is as follows:

SI&P octet: iiiiipxx
iiiii = SPDU identifier
p = parameter indication (always zero for CNS/ATM-1, indicating no parameters)
xx = parameters or special data field (always zero for CNS/ATM-1)

4.2.5 The short-form SPDUs, and their encoding for CNS/ATM-1 are as follows:

Table 4-1. Short Form Session Protocol Data Units

Value	Abbreviation	Full SPDU Name
E8	SCN	Short Connect
F8	SCNC	Short Connect Continue (**)
F0	SAC	Short Accept
D8	SACC	Short Accept Continue
E0	SRF	Short Refuse
A0	SRFC	Short Refuse Continue
88	SDT	Short Data Transfer (*)
B0	SAB	Short Abort (*)
C8	SFN	Short Finish (*)
D0	SDN	Short Disconnect (*)

(*) These SPDUs are not available when null-encoding is used in the data transfer phase, and are therefore not used for CNS/ATM-1.
(**) This SPDU is not used when using the short-connect protocol option to establish a session connection using the null-encoding option, and is therefore not used for CNS/ATM-1.

4.2.6 Mapping SPDUs to Transport Service primitives

4.2.7 The ATN upper layers make extensive use of the user-data capability of the transport service. The upper layers attempt to map the combined upper layer connection request information (comprising Session and Presentation Short Connect PDUs and ACSE AARQ, together with any user data) to the T-CONNECT service. If this is not possible, based on user-data size, then a transport connection is first established, using the T-CONNECT service, and the T-DATA service is then used to convey the upper layers connection information. 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 request, the D-START will be mapped to the T-CONNECT + T-DATA.

4.3 Use of the ATN Internet Transport Service

4.3.1 The use of the connection-oriented transport service provided by the ATN Internet is basically as specified in Clause 6 of the OSI CO Session Protocol definition (ISO/IEC 8327-1), with additions for ATN-specific features. Thus, the interface to the ATN Internet could be realised in an implementation by means of the standard XTI API, using the Options buffer for ATN-specific parameters.

4.3.2 The called and calling Transport Service Access Point (TSAP) address are provided to the TS-Provider on a per Transport Connection basis, using the called and calling Presentation Addresses as provided to ACSE in the A-ASSOCIATE request, with null presentation and session selectors.

4.3.3 The calling Presentation Address is known by local knowledge, while the called Presentation Address is obtained from a look-up table using the D-START Called Peer Id parameter.

4.3.4 Use of Transport Expedited Service

4.3.5 The TS-user indicates in all T-CONNECT requests that the transport expedited flow is not required.

4.3.6 Use of Transport Checksum / RER QoS parameter

4.3.7 SARPs 5.5.1.2 requires that the TS-user specifies the required residual error rate (RER) to determine whether or not the transport checksum is required. Information on the use or non-use of the transport checksum is conveyed between the TS-User and TS-Provider via the "residual error rate" component of the T-CONNECT Quality of Service (QoS) parameter.

4.3.8 In the ATN, the QoS provided to applications is maintained using capacity planning techniques that are outside of the scope of the SARPs. Network administrators are responsible for designing and implementing a network that will meet the QoS requirements of the CNS/ATM applications that use it.

4.3.9 If the TS-User requests the use of checksum (RER = "low") in the request primitive, the peer can only accept the use of checksum for this Transport Connection. If the TS-User proposes non-use of checksum (RER = "high") in the request primitive, the peer can either accept the non-use of checksum or force the use of checksum for this Transport Connection.

4.3.10 The use or non-use of the transport checksum is negotiated by the TS-Provider on a per Transport Connection basis, based on TS-User requests in the T-CONNECT request and response primitives, as follows:

a) If the required residual error rate in the T-CONNECT request has the abstract value "low", then the TS-provider uses best endeavours to obtain the lowest available residual error rate, including the use of the transport checksum in all Transport Protocol Data Units (TPDUs). The residual error rate in the T-CONNECT indication is set to the abstract value "low", and the responder can only accept this value in the T-CONNECT response.
b) If the required residual error rate in the T-CONNECT request has the abstract value "high", then the TS-provider proposes non-use of the transport checksum. The residual error rate in the T-CONNECT indication is set to the abstract value "high", and the responder can either accept this value, or request "low" in the T-CONNECT response. In the former case, transport checksum is not used, and in the latter case the TS-provider uses the transport checksum for all TPDUs.

4.3.11 Use of Priority parameters

4.3.12 Although transport priority and network priority are semantically independent of each other, it is required (in SARPs 5.5.1.2), that the TS-user specifies the Application Service Priority, which in turn is mapped into the resulting CLNP PDUs according to Table 1.3-2, which defines the fixed relationship between transport priority and the network priority. The Application Service Priority is provided to the TS-Provider on a per Transport Connection basis, via the TC priority quality of service parameter, using the values for Transport Layer Priority specified in Table 1.3-2.

4.3.13 Use of CLNP Security Label

4.3.14 The ATN Security Label is used to specify information about the traffic type and the class of communication.

4.3.15 It is provided to the TS-Provider on a per Transport Connection basis. It is conveyed by local means, using the encoding specified in 5.6.2.2.2. SARPs 5.2.7.3.1 states: "The mechanism by which the

[transport] connection initiator provides the appropriate ATN Security Label is a local matter. For example, it may be identified by an extension to the transport service interface, be implicit in the choice of a given TSAP, or be identified using a Systems Management function."

4.3.16 The D-START QoS parameter "Routing Class" is conveyed as the Security Tag field of the security tag set for Traffic Type and Associated Routing Policies within the ATN Security Label. For "General Communications" traffic, no security label is encoded.

4.3.17 SARPs 5.5.1.2 states that the TS-User provides the complete ATN Security Label (although only security tag value is of relevance). The encoding of the ATN Security Label is summarised below. It consists of all fields under the heading "Security Label" (i.e. 12 octets). The D-START QoS parameter "Routing Class" maps to the field labelled "Traffic Type & Category".

ATN Security Label Field	Value (Hex)	Length (Octets)
Security Format	C0	1
Security Label:		
Security Registration ID Length	06	1
Security Registration ID =		
OID {1.3.27.0.0}	06 04 2B 1B 00 00	6
Security Information Length	04	1
Security Information:		
Tag Set Name Length	01	1
Tag Set Name =		
"Traffic Type & Associated		
Routing Policies"	0F	1
Tag Set Length	01	1
Security Tag Value =:		
Traffic Type & category		
(from Table 5.6-1)	01 (for example)	1

5. PRESENTATION

5.1 Presentation Layer Functionality

5.1.1 The efficiency amendment to the presentation service defines the pass-through access to the session service, in particular the (new) NOR 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 optimisations available at the presentation layer are new protocol options, i.e., alternative, efficient PCI and procedures.

5.1.2 The efficiency amendment to the presentation protocol defines a number of 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. These are:

a) null-encoding protocol option
b) short-connect protocol option
c) short-encoding protocol option
d) nominated context protocol option
e) packed encoding protocol option

5.1.3 The first two of these options are selected for the CNS/ATM-1 profile, and are summarised briefly below.

5.1.4 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.

5.1.5 The *short-connect protocol option*. It is possible to use the short-connect option, which permits an efficient negotiation, during connection establishment, of the null-encoding protocol option, if both conditions a) and b) below are true:

a) the calling and called presentation selectors are null; and
b) the presentation-requirements parameter in the P-CONNECT service includes the kernel functional unit only.

5.1.6 The short-connect protocol option allows the negotiation of the encoding rules to 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.

5.2 Presentation Provider Abort Handling.

5.2.1 When using the null encoding presentation protocol option, then the ARP (Provider Abort) PPDU is not applicable, as specified in the ULA SARPs, Table 4.5-8. The ISO Presentation Protocol efficiency amendment states that if the null encoding protocol option has been selected, then the PPM issues an S-U-ABORT request primitive with no SS-user-data parameter, rather than sending an ARP PPDU.

5.2.2 Hence, an abnormal release by the Presentation Layer would encapsulate no user data in an S-U-ABORT request. The question therefore arises: when a S-U-ABORT Indication is received, how to distinguish whether the remote presentation entity experienced an abnormal release?

5.2.3 In fact, the PPM treats a received S-U-ABORT containing no User-data in the same way as it treats an S-U-ABORT with an embedded ARP, i.e. maps it to a P-P-ABORT indication primitive (see 6.4.4.6 in ISO/IEC 8823-1:1994/Amd.1:1997).

5.3 Presentation User Abort Handling.

5.3.1 The ACSE standard says that a P-U-ABORT indication with no User-data implies the existence of an ABRT APDU (i.e. a null-encoded ABRT with no parameters), and this is mapped by the ACSE protocol machine to an A-ABORT indication with Abort Source = "ACSE service-user" (8650-1 para 7.3.3.2.a) - NOT an A-P-ABORT ind.

5.4 Short PPDU Use and Encoding

5.4.1 The efficiency enhancements to the presentation protocol include the definition of "short" presentation protocol data units (PPDUs) which are distinguishable from the conventional longer form PPDUs.

5.4.2 The PCI of the Short PPDUs is a single octet encoded as follows:

0yyy00zz
zz = encoding choice (10 = unaligned PER)
yyy = Reason parameter:
000 presentation-user
001 reason not specified (transient)
010 temporary congestion (transient)
011 local limit exceeded (transient)
100 called presentation-address unknown (permanent)
101 protocol version not supported (permanent)
110 default context not supported (permanent)
111 user data not readable (permanent)

5.4.3 The short-form PPDUs, and their encoding for CNS/ATM-1 are as follows:

Table 5-1. Short Form Presentation Protocol Data Units

Value	Abbreviation	Full PPDU Name
02	SHORT-CP	Short Presentation Connect PPDU
02	SHORT-CPA	Short Presentation Connect Accept PPDU
x2	SHORT-CPR	Short Presentation Connect Reject PPDU(x = Reason)

5.4.4 This PCI is followed by the User-data, which is of type null-encoding.

5.4.5 On receipt of a S-CONNECT indication, the receiving PPM needs to determine whether or not the short-connect encoding option has been selected by the sending PPM. This determines the contents of the User Data parameter of the S-CONNECT indication primitive: either a (long form) CP or a SHORT-CP.

5.4.6 The SHORT-CP will always be 0000 00zz (and in our case zz will always be 10 - unaligned PER).

5.4.7 The CP will be encoded as follows:

a) in the case of BER-encoding, the first octet will be 31H (SET)
b) in the PER-encoding case, the first two bits will define which of the 2 optional set components are present. 8823-1 is not very clear, but it appears from 6.2.2.7 that the optional sequence containing the parameters for the CP must always be present in normal mode. Therefore the first 2 bits of the CP will be 01.

5.4.8 So the first 4 bits of the User Data parameter of the S-CONNECT indication are sufficient to identify the type of PPDU:

0000 - it is a SHORT-CP
0011 - it is a CP (BER-encoded)
0100 - it is a CP (PER-encoded)

5.4.9 In fact the ISO DAM text says in 6.2.6.8 "negotiation of [the PER] protocol option is not always possible. ... [Encoding the CP PPDU using PER] is only suitable if it is known by bilateral agreement that the PER protocol option is supported by the responder.

5.5 Guidance on PER Encoding of Character Strings.

5.5.1 OCTET STRING

5.5.2 As an example of a character string expressed in ASN.1, the CM SARPs defines the ASN.1 type rDP (routing domain part of the TSAP address) as OCTET STRING (SIZE(5)). Since this is a fixed-length type, no bits are needed in the PER encoding for the length of the string. For example, an rDP value of "AB901" (assuming ASCII encoding - actually the elements of the OCTET STRING are just binary values with no implicit interpretation) would be simply encoded as:

01000001	0x41 = "A"
01000010	0x42 = "B"
00111001	0x39 = "9"
00110000	0x30 = "0"
00110001	0x31 = "1"

5.5.3 IA5String (7-bit encoding)

5.5.4 As another example of a character string expressed in ASN.1, the CM SARPs has the following definition:

AircraftFlightIdentification ::= IA5String (SIZE(2..8))

5.5.5 The PER standard states that each IA5 character is encoded in the number of bits that is the smallest

that can accommodate all characters allowed by the effective PermittedAlphabet constraint (using the keyword FROM). IA5 does not have a PermittedAlphabet constraint, so the "effective permitted alphabet" is the entire alphabet.

5.5.6 IA5String is a "known-multiplier character string type" according to the PER standard. IA5String characters have values in the range 0..127. Therefore, in section 26.5.2 of the PER standards, N = 128, B = 7, B2 = 8 and b = 7). This means that, for UNALIGNED coding, each IA5String characters encodes into seven (7) bits.

5.5.7 Then the character string is encoded by concatenation of the 7-bit characters into a bit-field that is a multiple of 7 bits long. This is preceded by a length determinant field.

5.5.8 For example, a flight ID of "UA901" would be encoded in 38 bits as follows:

011	Length determinant: constrained length of IA5String = 5
1010101	"U"
1000001	"A"
0111001	"9"
0110000	"0"
0110001	"1"

5.5.9 Note that the use of unconstrained IA5Strings means that flight identifiers can contain upper and lower case characters as well as graphic characters (e.g. $%&*) and control characters (such as newline). For example, flight "BA 123" would not have the same encoding as flight "ba 123". It might be safer (and also more efficient in terms of encoding) to restrict the character set to numbers and upper case alphabetic.

5.5.10 Note that if the flight identification had been defined as

```
AircraftFlightIdentification ::= IA5String (SIZE(2..8)) FROM
("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ")
```

then the permitted alphabet contains 36 characters, and each character would be encoded into 6 bits.

5.6 Guidance on PER Encoding of Object Identifiers.

5.6.1 The ASN.1 type OBJECT IDENTIFIER is encoded in PER exactly the same as in BER. In PER, it is preceded by a length octet which gives the total number of octets in the encoded OID value. An OID value consists of a sequence of subidentifiers. The first two subidentifiers are combined into a single octet when encoded.

5.6.2 Subidentifiers in the range 0 - 127 are encoded as binary values in a single octet, with the most significant bit set to zero. Larger subidentifiers are encoded in a series of octets, such that the most significant bit of each octet indicates whether the are more octets to follow: it is set to zero in the last octet and one in each preceding octet.

5.6.3 An OID is always encoded into the minimum number of octets, so a subidentifier is not permitted to have leading zero values, i.e. the value 0x80 is not permitted as the most significant octet.

5.6.4 For example, for an aircraft whose 24-bit identifier is binary {0000 0001 1011 0110 0110 0110}, a Calling AP title value of {1.3.27.1.112230.0} would be encoded as follows:

0	no extension value present in calling-ap-title
0	indicates ap-title-form2 is used (i.e. Object Identifier form)
00000111	length of OID value = 7 octets
00101011	first two subidentifiers = {1.3}
00011011	third subidentifier = {27}
00000001	fourth subidentifier = {1}
10000110	m.s. bit = 1, continuation octet follows
11101100	m.s. bit = 1, continuation octet follows
01100110	m.s. bit = 0, fifth subidentifier = concatenation of 7-bit values
	0000110 1101100 1100110 = {112230}
00000000	sixth subidentifier = {0}

5.6.5 Note that the 24-bit aircraft identifier is actually encoded into 21 bits in this case. An end-system-id of 0x00 00 01 would be encoded in 1 octet, end-system-ids in the range 128 to 16,383 require 2 octets, and so on.

5.7 Guidance on encoding INTEGER types with discrete values

5.7.1 ASN.1 INTEGER types may be constrained such that their value is taken from a limited set of discrete values. For example, in the ACSE protocol, the following definition occurs:

Release-request-reason ::= INTEGER {normal (0), urgent (1), user-defined (30)} (0 | 1 | 30, ...)

5.7.2 The PER-visible constraints are given in parentheses, and make use of the extensibility notation to indicate that other values may be added in the future, ensuring backwards compatibility. In this example, only the values 0, 1 and 30 are possible if no extensions are present.

5.7.3 It might be expected that values of Release-request-reason would be encoded into 2 bits, as there are only three permitted values. However, the PER standard does not include this optimisation. The size of the encoding in the extension root is determined based only on the "range". The range is determined by the maximum and minimum permitted value, i.e. the upper bound (ub) and lower bound (lb), as (ub - lb + 1), so in this example it is 31. Therefore 5 bits are required to encode Release-request-reason.

5.7.4 Thus, the value "normal" would be encoded:

0 - extension bit: extension values not present
00000 - the range is 0 to 30, so 5 bits are needed to encode the value.

6. ACSE

6.1 ACSE Functionality

6.1.1 Application layer standards define the services and protocols provided 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 association control service element (ACSE) is one such common service. Others define the specific services and protocols supported by particular application layer standards.

6.1.2 The ACSE service allows one application-specific AE to request that an association be established with a remote AE 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, although it assumes that the session layer provides the orderly release capability, which is not the case in the profile.

6.1.3 The ACSE service and connection-oriented protocol are defined in ISO/IEC 8649 and ISO/IEC 8650-1 respectively. The CNS/ATM-1 profile is based on edition 2 of these standards, which define the following ACSE functional units:

a) Kernel
b) Authentication
c) Application Context Negotiation

6.1.4 The kernel and, optionally, the authentication FUs are selected in the CNS/ATM-1 profile.

6.2 Discussion of differences in ACSE editions

6.2.1 As ACSE is potentially available as commercial off-the-shelf software, a description is provided of the evolution of the ACSE standard. The editor’s preface to ISO/IEC 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 travelling on. For the ATN ULA, where there is zero session layer functionality in the data transfer phase, none of this matters, since there are no external events. A classic implementation, though, must put a appropriate EXTERNAL hooks in its state machine when things go wrong in classical session / presentation layers.

6.2.2 The changes in the ACSE protocol specification are roughly similar. ISO/IEC 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. For TC2, “A-RELEASE procedure is disrupted if P-RESYNCHRONIZE, P-U-EXCEPTION-REPORT, or P-EXCEPTION-REPORT primitives occur on the association”, which will never be the case for CNS/ATM-1.

6.2.3 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.

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

6.2.5 The changes to ACSE that are required are the requirements to encode the ACSE PDUs in PER, and to map the P-RELEASE primitives to P-DATA (this is done in the CF), and the addition of PER-visible extensibility markers.

6.3 Authentication Support

6.3.1 The ULCS SARPs includes optional limited support of the Authentication functional unit of ACSE (A-FU(AU)). The authentication parameters are not present if A-FU(AU) is not negotiated. The ATN specification is non-conformant to the ISO protocol requirements in that the Authentication Mechanism Name is not supported even when the Authentication FU is selected.

6.3.2 The problem arises from the complex ASN.1 definition of Authentication-mechanism-name, a parameter which is not required for the initial ATN realisation of "security hooks". An implementation would only implement A-FU(AU) if it supports an application which uses the D-START Security parameter (none of the ATN applications defined in SARPs currently do). The syntax of the optional authentication value is restricted to the ASN.1 types GraphicString, BIT STRING or EXTERNAL.

6.4 ACSE Definitions

6.4.1 The ACSE abstract syntax for CNS/ATM-1 is defined in ISO/IEC 8650-1:1995,ed.2/AM1. It is reproduced here for ease of reference. In case of any discrepancy, the ISO/IEC standard takes precedence. The elements new to the ACSE edition 2 with the ASN.1 extensibility notation (ISO/IEC 8650-1:1995,ed.2/AM1) of the connection-oriented ACSE are indicated by redlining.

```
ACSE-1 { joint-iso-itu-t association-control(2) modules(0) apdus(0) version1(1) }
-- ACSE-1 refers to ACSE version 1

DEFINITIONS ::=

BEGIN

EXPORTS
      acse-as-id, ACSE-apdu,
     aCSE-id, Application-context-name,
      AP-title, AE-qualifier,
      AE-title, AP-invocation-identifier,
      AE-invocation-identifier,
      Mechanism-name, Authentication-value,
      ACSE-requirements;

IMPORTS  Name, RelativeDistinguishedName
            FROM InformationFramework
                 { joint-iso-ccitt ds(5) module(1) informationFramework(1)  2 };
-- The data types Name and RelativeDistinguishedName are imported from ISO/IEC 9594-2.
-- object identifier assignments

acse-as-id OBJECT IDENTIFIER ::=
  { joint-iso-itu-t association-control(2) abstract-syntax(1) apdus(0)  version1(1) }
-- may be used to reference the abstract syntax of the ACSE APDUs
```

```
aCSE-id OBJECT IDENTIFIER ::=
  { joint-iso-itu-t association-control(2) ase-id(3) acse-ase(1) version(1) }
-- may be used to identify the Association Control ASE.

-- top level CHOICE

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 functional unit is selected.
    mechanism-name                  [11] IMPLICIT Mechanism-name
                                                 OPTIONAL,
   --   The following field shall only be present if the Authentication functional unit is selected.
  calling-authentication-value  [12]        EXPLICIT Authentication-value
                                                 OPTIONAL,
  application-context-name-list             [13]  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,
...,...,
    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,
```

```
        -- This 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,
        ..., ...,
          user-information             [30]    IMPLICIT Association-information
                                                              OPTIONAL
        }

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

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

      ABRT-apdu ::= [ APPLICATION 4 ] IMPLICIT SEQUENCE
        { abort-source                 [0]     IMPLICIT ABRT-source,
          abort-diagnostic             [1]     IMPLICIT ABRT-diagnostic OPTIONAL,
      --  This field shall not be present if only the Kernel is used.
        ..., ...,
          user-information             [30]    IMPLICIT Association-information  OPTIONAL
        }

        ABRT-diagnostic ::= ENUMERATED
         { no-reason-given (1),
           protocol-error (2),
           authentication-mechanism-name-not-recognized (3),
           authentication-mechanism-name-required (4),
           authentication-failure (5),
           authentication-required (6),
        ...
         }

     ABRT-source  ::=  INTEGER { acse-service-user (0), acse-service-provider (1)} (0..1, ...)

        ACSE-requirements  ::=  BIT STRING
         { authentication (0), application-context-negotiation(1) }

        Application-context-name-list ::= SEQUENCE OF Application-context-name

        Application-context-name ::= OBJECT IDENTIFIER
      --  Application-entity title productions follow (not in alphabetical order
     AP-title ::= CHOICE {
                    ap-title-form1      AP-title-form1,
                    ap-title-form2      AP-title-form2,
                    ... }

        AE-qualifier ::= CHOICE {
```

```
                ae-qualifier-form1              AE-qualifier-form1,
                ae-qualifier-form2              AE-qualifier-form2,
                ...     }
-- When both AP-title and AE-qualifier data values are present in an AARQ or AARE APDU, both must
-- have the same form to allow the construction of an AE-title as discussed in CCITT Rec. X.665 |
-- ISO/IEC 9834-6.

        AP-title-form1 ::= Name
        -- The value assigned to AP-title-form1 is The Directory Name of an application-process title.

        AE-qualifier-form1 ::= RelativeDistinguishedName
        -- The value assigned to AE-qualifier-form1 is the relative distinguished name of a particular
        - application-entity of the application-process identified by AP-title-form1.

        AP-title-form2 ::= OBJECT IDENTIFIER

        AE-qualifier-form2 ::= INTEGER
    AE-title ::= CHOICE {
                ae-title-form1      AE-title-form1,
                ae-title-form2      AE-title-form2,
                ...     }
-- As defined in CCITT Rec. X.650 | ISO 7498-3, an application-entity title is composed of an application-
-- process title and an application-entity qualifier. The ACSE protocol provides for the transfer of an
-- application-entity title value by the transfer of its component values. However, the following data type
-- is provided for International Standards that reference a single syntactic structure for AE titles.

        AE-title-form1 ::= Name
--  For access to The Directory (ITU-T Rec. X.500-Series | ISO/IEC 9594), an AE title has AE-title-form1.
--  This value can be constructed from AP-title-form1 and AE-qualifier-form1 values contained in an
--  AARQ or AARE APDU. A discussion of forming an AE-title-form1 from AP-title-form1 and AE-qualifier-
--  form1 may be found in CCITT Rec. X.665 | ISO/IEC 9834-6.

        AE-title-form2 ::= OBJECT IDENTIFIER
        - A discussion of forming an AE-title-form2 from AP-title-form2 and AE-qualifier-form2 may be
        -- found in CCITT Rec. X.665 | ISO/IEC 9834-6.

        AE-invocation-identifier ::= INTEGER

        AP-invocation-identifier ::= INTEGER
        --      End of Application-entity title productions

        Associate-result  ::= INTEGER
              { accepted (0),
              rejected-permanent (1),
              rejected-transient (2)
              } (0..2, ...)
        Associate-source-diagnostic  ::=  CHOICE
           { acse-service-user       [1] INTEGER
                                         { null (0),
                                          no-reason-given (1),
                                         application-context-name-not-supported (2),
                                         calling-AP-title-not-recognized (3),
                                         calling-AP-invocation-identifier-not-recognized (4),
                                         calling-AE-qualifier-not-recognized (5),
                                         calling-AE-invocation-identifier-not-recognized (6),
                                         called-AP-title-not-recognized (7),
                                         called-AP-invocation-identifier-not-recognized (8),
                                         called-AE-qualifier-not-recognized (9),
                                         called-AE-invocation-identifier-not-recognized (10),
                                         authentication-mechanism-name-not-recognized (11),
                                         authentication-mechanism-name-required (12),
                                         authentication-failure (13),
```

```
                                        authentication-required (14)
                                    } (0..14 ,  ...),
            acse-service-provider [2] INTEGER
                                        { null (0),
                                        no-reason-given (1),
                                        no-common-acse-version (2)
                                        }(0..2 , ...)
        }
      Association-information ::=  SEQUENCE SIZE (1, ..., 0 | 2..MAX) OF EXTERNAL

          Authentication-value  ::=   CHOICE
           { charstring [0] IMPLICIT GraphicString,
             bitstring    [1] IMPLICIT BIT STRING,
             external     [2] IMPLICIT EXTERNAL,
             other        [3] IMPLICIT SEQUENCE {
                    other-mechanism-name  MECHANISM-NAME.&id ({ObjectSet}),
                    other-mechanism-value  MECHANISM-NAME.&Type ({ObjectSet}{@.other-mechanism-
name})
                                                    }
          }
--    The abstract syntax of (calling/responding) authentication-value is determined by the authentication
--    mechanism used during association establishment.  The authentication mechanism is either explicitly
--    denoted by the &id field (of type OBJECT IDENTIFIER) for a mechanism belonging to the class
-- MECHANISM-NAME, or it is known implicitly by
--    prior agreement between the communicating partners.  If the "other" component is chosen, then
-- the "mechanism-name" component must be present in accordance with
--  ITU-T Rec. X.680|ISO/IEC 8824. If the value "mechanism-name" occurs in the AARQ-apdu or the
-- AARE-apdu, then that value must be the same as the value for "other-mechanism-name"

          Implementation-data ::=  GraphicString

          Mechanism-name ::= OBJECT IDENTIFIER

      MECHANISM-NAME ::=TYPE-IDENTIFIER
      ObjectSet MECHANISM-NAME ::= {...}

       Release-request-reason  ::=  INTEGER
                          { normal (0) , urgent (1) ,  user-defined (30) } (0 | 1 | 30, ...)

      Release-response-reason  ::=  INTEGER
                          { normal (0) , not-finished (1) ,  user-defined (30) } (0 | 1 | 30, ...)

          END
```

6.5 ACSE Encoding Guidance

6.5.1 Extensibility

6.5.2 The internationally agreed version of the abstract syntax for ACSE edition 2 in ISO/IEC 8650-1 Amendment 1 includes some new ASN.1 features. One such feature is the *extension marker pair* notation (... , ...), which is used in the ACSE APDUs. The feature is the subject of a draft technical corrigendum to the ASN.1 standard (ISO/IEC 8824-1/Cor 1).

6.5.3 The new ASN.1 feature was introduced by the ISO committees as it was felt desirable to allow extensions to be included in the middle of a SEQUENCE. Previously, only a single extension marker could

be present as the last item in an ASN.1 construction. Now, this is treated as a special case and when a single extension marker appears as the last item in a type, a matching extension marker is assumed to exist just before the closing brace of the type. Extension additions will always be made between pairs of extension markers. The "extensions SEQUENCE { ... } OPTIONAL" construction was felt to be too cumbersome and not completely general purpose. It offended the ASN.1 purists, who felt this was a real deficiency and so raised the corrigendum on ISO/IEC 8824-1. Also, there was a danger of producing a conformant BER-encoded ACSE which is not backwards-compatible.

6.5.4 The ISO committee who were adding extensibility to ACSE edition 2 thought that it would be clearer to use this new feature of ASN.1. The "value added" is a more pure ASN.1 which should be easier to read and to understand why the extensions field is present.

6.5.5 ISO/IEC 8824-1/Cor 1 has been written so that the encoding of extensibility as already described in ISO/IEC 8825-2 remains correct.

6.5.6 A SEQUENCE containing the extension marker pair is encoded with one bit in the preamble to indicate whether or not extensions are present. If not, the bit is set to zero, and nothing is encoded for the extensions field. If extensions are present, the bit in the preamble is set to one and the SEQUENCE is encoded accordingly.

6.5.7 This differs from previous drafts of ACSE extensibility amendment, which used the following construction:

extensions SEQUENCE { ... } OPTIONAL,

6.5.8 While semantically equivalent, the more recent extension marker pair notation is not encoded identically to the above notation, which had an OPTIONAL member of the SEQUENCE which then contained the extensibility marker. In that case, the bitmap in the preamble (instead of the extensibility bit) indicated whether or not any extensions were present. The encoding overhead is identical in both cases if no extensions are present, and differs very slightly if extensions are present.

6.5.9 The CNS/ATM-1 profile requires the correct encoding and decoding of the recent ASN.1 extension marker pair notation.

6.5.10 Encoding of ACSE User Information

6.5.11 The ACSE abstract syntax contains the definition:

Association-information ::= SEQUENCE SIZE (1, ..., 0 | 2 .. MAX) OF EXTERNAL

6.5.12 User data in D-START request and response primitives is mapped by ACSE onto the "user-information" field of AARQ and AARE APDUs respectively. Similarly, User data in D-END request and response primitives is mapped by ACSE onto the user-information field of RLRQ and RLRE APDUs respectively.

6.5.13 The Association-information type which is used for the user-information field in ACSE APDUs has a size constraint SIZE (1, ..., 0 | 2 .. MAX). This means that in most cases the value of the extensibility marker is expected to default to zero (i.e. "no extensions present") and the type will be encoded as SIZE (1), so no bits are required to encode the size. If the extensibility marker is set to 1, then the size can

additionally either be zero, or in the range 2 .. MAX (i.e. effectively unconstrained). For CNS/ATM-1, the size will always be 1, and the extensibility marker will therefore always be set to 0.

6.5.14 The PER standard defines EXTERNAL as:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE {
        direct-reference            OBJECT IDENTIFIER   OPTIONAL,
        indirect-referenceINTEGER                       OPTIONAL,
        data-value-descriptor       ObjectDescriptorOPTIONAL,
        encoding            CHOICE {
                single-ASN1-type        [0] ABSTRACT-SYNTAX.&Type,
                octet-aligned           [1] IMPLICIT OCTET STRING,
                arbitrary               [2] IMPLICIT BIT STRING }}
```

6.5.15 When PER is used, the definition of the encoding of the EXTERNAL type thus allows several different optional elements to be present and encoding choices to be taken. ULCS SARPs 4.6.6.3.3 defines a restriction of the general case for EXTERNAL encoding in ACSE APDUs (this was introduced in UL-DR 115; before this DR, all implementations were required to handle all possible types of encoding for both sending and receiving. This was considered an onerous requirement, and a possible source of interworking failures).

6.5.16 The “single-ASN1-type” seems appropriate, as it allows the presentation context of the application data to be explicitly identified (for future extensibility) by means of the “indirect-reference”, and it avoids the need to encode the length determinant of the bit string comprising the application data.

6.5.17 Thus, the ULCS SARPs allows either the single-ASN1-type or the arbitrary (BIT STRING) choice when encoding user-information. For decoding, support for both forms is required. The octet-aligned form is out of the scope of the ULCS SARPs.

6.5.18 When the single-ASN1-type encoding form is used, the indirect-reference field contains a presentation-context-id value as specified in ULCS SARPs Table 4.3-3.

6.5.19 When the arbitrary (BIT STRING) encoding form is used, the indirect-reference field is absent.

6.5.20 The direct-reference and data-value-descriptor fields are not used in the CNS/ATM-1 profile.

6.6 Examples of ACSE encoding

6.6.1 In the following examples of APDU encoding, the ASN.1 record is first presented to show the actual values being encoded. This is followed by a hexadecimal view of the encoded data, and a binary view which explains the encoding in detail. To make it easier to read the binary view of the data, blank lines are used to group fields that logically belong together (typically length/value pairs) ; a newline is used to delineate fields ; space is used to delineate characters within a character string ; a period (.) is used to mark octet boundaries ; and an ‘x’ represents a zero-bit used to pad the final octet to an octet boundary.

6.6.2 ACSE Associate Request (AARQ) APDU

ASN.1 Record

```
ACSE-apdu
{AARQ-apdu
        {
          application-context-name "{1 3 27 3 1}",
          calling-AP-title "{1 3 27 1 500 0}"
          calling-AE-qualifier "1 (CM)"
          user-information }
}
```

Hexadecimal view (17 octets up to start of user information)

00 30 10 42 B1 B0 30 10 18 AC 6C 06 0D D0 00 01 01 ...

Binary view

0	Extension bit: No extension values present in ACSE-apdu
000	Indicates AARQ-apdu is used (first item in ACSE-apdu CHOICE)
0 000.0011 0000.0001	Extension bit: No extension value present in AARQ-apdu Bitmap indicates presence of the following OPTIONAL fields: Calling AP Title, Calling AE Qualifier, user-information
0000.0100 0010.1011 0001.1011 0000.0011 0000.0001	Length of Application context name = 4 octets Application Context Name = {1.3.27.3.1 } : Version = 1
0 0 00.000110 00.101011 00.011011 00.000001 10.000011 01.110100 00.000000	No extension value present in Calling AP-Title Indicates AP-title-form2 is used Length of Calling AP-Title = 6 Calling AP-Title = {1 3 27 1 500 0}
0 0. 00000001. 00000001.	No extension values present in Calling AE-qualifier Indicates AE-qualifier-form2 is used Length of Calling AE-qualifier = 1 Calling AE-qualifier = 1 (CMA)
0 000 10 10.0000 0011.1100 00 etc.	No extension values present in user-information, so SEQUENCE OF is exactly 1 in length Bitmap indicates presence of no OPTIONAL fields in EXTERNAL type Choice 2 = BIT STRING encoding Length determinant for bit string = F0 = 240 bits (dec) User information follows (encoded according to application SARPs) ...

6.6.3 ACSE Associate Response (AARE) APDU

ASN.1 Record

```
ACSE-apdu
{AARE-apdu
        {  application-context-name "{1 3 27 3 1}",
          result "accepted (0)"
          result-source-diagnostic
                {
```

```
                    acse-service-user "null (0)"
                  }
        user-information }
}
```

Hexadecimal view (9 octets up to start of user-information)

10 01 04 2B 1B 03 01 00 05 ...

Binary view

0	Extension bit: No extension values present in ACSE-apdu
001	Indicates AARE-apdu is used (second item in ACSE-apdu CHOICE)
0 000.0000 0001.	Extension bit: No extension value present in AARE-apdu Bitmap indicates presence of the following OPTIONAL fields: user-information
0000 0100. 00101011. 00011011. 00000011. 00000001.	Length of Application context name = 4 Application Context Name OID = {1 3 27 3 1 }
0 00	Extension bit: No extension values present in Associate-result Result = 0 (accepted)
0	Indicates acse-service-user CHOICE is used in Associate Source Diagnostic
0 000.0	No extension values present in Source Diagnostic Source Diagnostic = 0 (null)
0	No extension values present in user-information, so SEQUENCE OF is exactly 1 in length
000	Bitmap indicates presence of no OPTIONAL fields in EXTERNAL type
10 1.000 0000 1.010 1100 etc.	Choice 2 = BIT STRING encoding Length determinant for bit string = 0AC = 172 bits (dec) User information follows (encoded according to application SARPs) ...

6.6.4 ACSE Release Request (RLRQ) APDU

ASN.1 Record

```
ACSE-apdu
{RLRQ-apdu {
        user-information }
}
```

Hexadecimal view

26 02 ...

Binary view

0	Extension bit: No extension values present in ACSE-apdu
010	Indicates RLRQ-apdu is used (third item in ACSE-apdu CHOICE)
0 11	Extension bit: No extension value present in RLRQ-apdu Bitmap indicates presence of the following OPTIONAL fields: Release-request-reason, user-information

0.	Extension bit: No extension values present in Release-request-reason
00	Result = 0 (normal)
0	No extension values present in user-information, so SEQUENCE OF is exactly 1 in length
000	Bitmap indicates presence of no OPTIONAL fields in EXTERNAL type
10.	Choice 2 = BIT STRING encoding
0001 1100	Length determinant for bit string = 1C = 28 bits (dec)
etc.	User information follows (encoded according to application SARPs) ...

6.6.5 ACSE Release Response (RLRE) APDU

ASN.1 Record

```
ACSE-apdu
{RLRE-apdu {
        user-information }
}
```

Hexadecimal view

36 02 ...

Binary view

0	Extension bit: No extension values present in ACSE-apdu
011	Indicates RLRE-apdu is used (fourth item in ACSE-apdu CHOICE)
0	Extension bit: No extension value present in RLRE-apdu
11	Bitmap indicates presence of the following OPTIONAL fields: Release-response-reason, user-information
0.	Extension bit: No extension values present in Release-response-reason
00	Result = 0 (normal)
0	No extension values present in user-information, so SEQUENCE OF is exactly 1 in length
000	Bitmap indicates presence of no OPTIONAL fields in EXTERNAL type
10.	Choice 2 = BIT STRING encoding
0001 1100	Length determinant for bit string = 1C = 28 bits (dec)
etc.	User information follows (encoded according to application SARPs) ...

7. PER ENCODING EXAMPLES

7.1 Purpose

7.1.1 This section contains examples of complete APDUs for the CNS/ATM-1 applications. Thus, it illustrates the complete user-information fields in the ACSE APDUs given in the previous chapter, as well as examples of P-DATA encodings and Presentation and Session Short PDUs.

7.2 CM Logon Request sent from Air to Ground

7.2.1 This example illustrates how the CM Logon Request PDU is sent as the User-information field of an ACSE A-Associate Request APDU. This is an example of an acse-apdu being sent on P-CONNECT (therefore, it is not encoded as Fully-encoded-data, which only applies in the data transfer phase).

ASN.1 Record

```
ACSE-apdu
{AARQ-apdu
        {
         application-context-name "{1 3 27 3 1}",
         calling-AP-title "{1 3 27 1 500 0}"
         calling-AE-qualifier "1 (CM)"
         user-information }
}
CMAircraftMessage
{CMLogonRequest
        {
         aircraftFlightIdentification “{UA901}”
         cMLongTSAP
                {
                 rDP “{AB901}”
                 shortTsap
                        { locSysNselTsel “{4440900901}” }
                }
        facilityDesignation “{KIADIZDS}”
}
```

Hexadecimal view (51 octets)

E8 02 00 30 10 42 B1 B0 30 10 18 AC 6C 06 0D D0 00 01 01 0A 03 90 10 EA C1 72 C1 8A 0A 11 C9 81 88 68 68 68 60 72 60 60 72 60 63 25 C9 83 12 4D A8 94 C0

Binary view

1110 1000.	Session SCN SPDU
0000 0010.	Presentation Short-CP PPDU(indicates unaligned PER) ACSE APDU
0	Extension bit: No extension values present in ACSE-apdu
000	APDU is an AARQ (first item in ACSE-apdu CHOICE)
0	Extension bit: No extension value present in AARQ-apdu

000.0011 0000.0001	Bitmap indicates presence of the following OPTIONAL fields: Calling AP Title, Calling AE Qualifier, user-information
0000.0100	Length of Application context name = 4 octets
0010.1011 0001.1011 0000.0011 0000.0001	Application Context Name = {1.3.27.3.1 } : Version = 1
0	No extension value present in Calling AP-Title
0	Indicates AP-title-form2 (Object Identifier form) is used
00.0001 10	Length of Calling AP-Title = 6
00.1010 1100.0110 1100.0000 0110.0000 1101.1101 0000.0000 00	Calling AP-Title = {1 3 27 1 500 0}
0	No extension values present in Calling AE-qualifier
0.	Indicates AE-qualifier-form2 (Object Identifier form) is used
0000 0001.	Length of Calling AE-qualifier = 1
0000 0001.	Calling AE-qualifier = 1 : corresponds to CMA
	user-information
0	Extension bit: No extension values present in user-information, so SEQUENCE OF is exactly 1 in length
000	Bitmap indicates no OPTIONAL fields present in EXTERNAL type
10	Choice 2 = BIT STRING encoding
10.0000 0011.1001 00	Length determinant for bit string = E4 = 228 bits (dec)
	CM Logon PDU
0	Extension bit: no extensions in CMAircraftMessage
0.0	CHOICE 0 in CMAircraftMessage = CMLogonRequest
001 000	bit map - only OPTIONAL field present is FacilityDesignation
	aircraftFlightIdentification IA5String SIZE(2..8)):
0.11	constrained length of IA5String = 5
10 1010.1	"U"
100 0001.	"A"
0111 001	"9"
0.1100 00	"0"
01.1000 1	"1"
	cMLongTSAP ::= SEQUENCE
	rDP OCTET STRING (SIZE(5))
010.0000 1	"A"
010.0001 0	"B"
001.1100 1	"9"
001.1000 0	"0"
001.1000 1	"1"
	shortTsap
0	bitmap - OPTIONAL aRS field is absent
	locSysNselTsel: OCTET STRING (SIZE(10..11))
0	constrained length of locSysNselTsel = 10
0.0110 100	"4"
0.0110 100	"4"
0.0110 100	"4"
0.0110 000	"0"
0.0111 001	"9"
0.0110 000	"0"
0.0110 000	"0"
0.0111 001	"9"
0.0110 000	"0"
0.0110 001	"1"
	FacilityDesignation ::= IA5String (SIZE(4..8))
1.00	constrained length of facilityDesignation = 8
10 0101.1	"K"
100 10 01.	"I"
1000 001	"A"
1.0001 00	"D"
10.0100 1	"I"
101.1010	"Z"
1000.100	"D"
1 0100.11	"S"
00 0000.	Padding bits

7.3 CM Logon (maintain) response sent from Ground to Air

7.3.1 This example illustrates how the CM Logon Response PDU is sent as the User-information field of an ACSE A-Associate Response APDU. This is another example of an acse-apdu being sent on P-CONNECT (therefore, it is not encoded as Fully-encoded-data, which only applies in the data transfer phase).

ASN.1 Record

```
ACSE-apdu
{AARE-apdu
        {  application-context-name "{1 3 27 3 1}",
          result "accepted (0)"
          result-source-diagnostic
                {
                  acse-service-user "null (0)"
                }
          user-information }
}
CMGroundMessage
{CMLogonResponse
        {
          airInitiatedApplications
                {
                  aeQualifier “{2}” (CPC)
                  apVersion “{1}”
                  apAddress
                        {
                          shortTsap
                                { locSysNselTsel “{Gnd1SystTWO }” }
                        }
                }
          groundOnlyInitiatedApplications
                {
                  aeQualifier “{0}” (ADS)
                  apVersion “{1}”
                }
        }
}
```

Hexadecimal view (31 octets)

F0 02 10 01 04 2B 1B 03 01 00 05 01 22 18 00 10 05 47 6E 64 31 53 79 73 74 54 57 4F 00 02 00

Binary view

1111 0000.	Session SAC SPDU
0000 0010.	Presentation Short-CPA PPDU (indicates unaligned PER) ACSE APDU
0	Extension bit: No extension values present in ACSE-apdu

001	APDU is an AARE (second item in ACSE-apdu CHOICE)
0	Extension bit: No extension value present in AARE-apdu
000.0000 0001.	Bitmap indicates presence of the following OPTIONAL fields: user-information
0000 0100.	Length of Application context name = 4
0010 1011. 0001 1011. 0000 0011. 0000 0001.	Application Context Name OID = {1 3 27 3 1 }
0	Extension bit: No extension values present in Associate-result
00	Result = 0 (accepted)
0	Indicates acse-service-user CHOICE is used in Associate Source Diagnostic
0	No extension values present in Associate Source Diagnostic
000.0	Associate Source Diagnostic = 0 (null)
	user-information
0	Extension bit: No extension values present in user-information, so SEQUENCE OF is exactly 1 in length
00 0	Bitmap indicates no OPTIONAL fields present in EXTERNAL type
10	Choice 2 = BIT STRING encoding
1.0000 0001.0010 001	Length determinant for bit string = 091 = 145 bits (dec)
	CM Logon PDU
	CMGroundMessage ::= CHOICE
0.	no extensions
000	choice 0 = CMLogonResponse
	CMLogonResponse ::= SEQUENCE
1 1	Bitmap indicates presence of :airInitiatedApplications, groundOnlyInitiatedApplications
000.0000 0	airInitiatedApplications: Size of SEQUENCE OF = 1
000.0001 0	AEQualifier ::= INTEGER (0..255), value = 2 (CPC)
000.0000 0	apversion : VersionNumber ::= INTEGER (1..255), constrained value = 1
1	APAddress ::= CHOICE, choice 1 : ShortTsap
	shortTsap ::= SEQUENCE
0	Bitmap - OPTIONAL aRS field is absent
	locSysNselTsel: OCTET STRING (SIZE(10..11))
1.	constrained length of locSysNselTsel = 11
0100 0111.	"G"
0110 1110.	"n"
0110 0100.	"d"
0011 0001.	"1"
0101 0011.	"S"
0111 1001.	"y"
0111 0011.	"s"
0111 0100.	"t"
0101 0100.	"T"
0101 0111.	"W"
0100 1111.	"O"
0000 0000.	groundOnlyInitiatedApplications: Size of SEQUENCE OF = 1
0000 0010.	AEQualifier ::= INTEGER (0..255), value = 0 (ADS)
0000 0000.	apversion : VersionNumber ::= INTEGER (1..255), constrained value = 1

7.4 CM End request sent from Ground to Air

7.4.1 In this example, a dialogue has previously been established, and the CM-End service is invoked when in data transfer phase. Therefore, as specified in ULCS SARPs 4.3.2.6.2, the presentation User Data is encoded as Fully-encoded-data.

7.4.2 This is an example of an acse-apdu being sent on P-DATA.

ASN.1 Record

ACSE-apdu
{RLRQ-apdu {
Release-request-reason }
}
CMGroundMessage - none. (The CM End request is mapped to D-END request by the CM-ground-ASE, with no user data).

Hexadecimal view (4 octets)

00 20 A1 40

Binary view

	T-DATA User data
	Fully-encoded-data ::= SEQUENCE SIZE (1, ...) OF PDV-list
0	Extension bit: no extensions, therefore 1 element in SEQUENCE OF
	PDV-list ::= SEQUENCE
0	Bitmap - no optional elements in PDV-list
	Presentation-context-identifier ::= INTEGER (1..127, ...)
0	Extension bit: no extensions, therefore size is constrained to 1 .. 127
0 0000.00	Constrained value = 1 (acse-apdu)
10	choice 2 = arbitrary (BIT STRING) encoding
0000.1010	length determinant = 0AH= 10 (dec) bits
	ACSE APDU
0	Extension bit: No extension values present in ACSE-apdu
010.	Indicates RLRQ-apdu is used (third item in ACSE-apdu CHOICE)
0	Extension bit: No extension value present in RLRQ-apdu
10	Bitmap indicates presence of the following OPTIONAL fields: reason
	reason : Release-request-reason ::= INTEGER (0|1|30, ...)
0	Extension bit: No extension values present
00	value = 0 (normal)
00.	Padding bits

7.5 D-END Response generated by CM-Air-ASE

7.5.1 This is an example of an acse-apdu being sent on P-DATA. Again, the presentation User Data is encoded as Fully-encoded-data.

ASN.1 Record

ACSE-apdu
{RLRE-apdu {
user-information }
}
CMAirMessage - none. (The CM-air-ASE generates a D-END response with no user data).

Hexadecimal view (4 octets)

00 20 A3 40

Binary view

	T-DATA User data
	Fully-encoded-data ::= SEQUENCE SIZE (1, ...) OF PDV-list
0	Extension bit: no extensions, therefore 1 element in SEQUENCE OF
	PDV-list ::= SEQUENCE
0	Bitmap - no optional elements in PDV-list
	Presentation-context-identifier ::= INTEGER (1..127, ...)
0	Extension bit: no extensions, therefore size is constrained to 1 .. 127
0 0000.00	Constrained value = 1 (acse-apdu)
10	choice 2 = arbitrary (BIT STRING) encoding
0000.1010	length determinant = 0AH= 10 (dec) bits
	ACSE APDU
0	Extension bit: No extension values present in ACSE-apdu
011.	Indicates RLRE-apdu is used (fourth item in ACSE-apdu CHOICE)
0	Extension bit: No extension value present in RLRE-apdu
10	Bitmap indicates presence of the following OPTIONAL fields: reason
	reason : Release-response-reason ::= INTEGER (0|1|30, ...)
0	Extension bit: No extension values present
00	value = 0 (normal)
00.	Padding bits

7.6 ADS Demand Contract Request, existing dialogue

7.6.1 This example illustrates the encoding of an ADS demand contract. The example assumes that a dialogue has previously been established between ground and air systems (e.g. by means of an event contract), so that the demand contract request is sent using the D-DATA rather than the D-START service. Thus, the encoded ADS APDU is mapped to P-DATA User Data, encoded as Fully-encoded-data.

7.6.2 This is an example of a user-ase-apdu being sent on P-DATA.

ASN.1 Record

ACSE-apdu - none. (The association already exists and the CF is in DATA TRANSFER state).

```
ADSGroundPDUs
{ aDS-demand-contract-PDU
        {
         aircraft-address (NULL)
         projected-profile (NULL)
         ground-vector (NULL)
         air-vector (NULL)
         short-term-intent
                {
                 projectionTime “{10}” (minutes)
                }
         extended-projected-profile
                {
                 number-of-way-points “{2}”
                }
        }
}
```

Hexadecimal view (6 octets)

00 A1 C3 7B 09 81

Binary view

	T-DATA User data
	Fully-encoded-data ::= SEQUENCE SIZE (1, ...) OF PDV-list
0	Extension bit: no extensions, therefore 1 element in SEQUENCE OF
	PDV-list ::= SEQUENCE
0	Bitmap - no optional elements in PDV-list
	Presentation-context-identifier ::= INTEGER (1..127, ...)
0	Extension bit: no extensions, therefore size is constrained to 1 .. 127
0 0000.10	Constrained value = 3 (user-ase-apdu)
10	choice 2 = arbitrary (BIT STRING) encoding
0001.1100	length determinant = 1CH= 28 (dec) bits
	ADS APDU
0	Extension bit: No extension values present in ACSE-apdu
011.	Indicates aDS-demand-contract-PDU (fourth item in ADSGroundPDUs CHOICE)
0	Extension bit: No extension value present in DemandContract
111 1011.	Bitmap - indicates presence of: aircraft-address, projected-profile, ground-vector, air-vector, short-term-intent, extended-projected-profile.
	aircraft-address (NULL)
	projected-profile (NULL)
	ground-vector (NULL)
	air-vector (NULL)
0000 1001.	short-term-intent: ProjectionTime ::= INTEGER (1..240), constrained value = 10 (minutes)
	extended-projected-profile: ExtendedProjectedProfileRequest
1	CHOICE 1 = number-of-way-points
000 0001.	INTEGER (1..128), constrained value = 2

7.7 CPDLC DSC END REQUEST

7.7.1 This example illustrates both an acse-apdu and an embedded application-apdu being sent over an existing connection using the T-DATA service. The presentation User Data is encoded as Fully-encoded-data.

ASN.1 Record

```
ACSE-apdu
{RLRQ-apdu {
          reason "{ 0 }" (normal)
          user-information }
}
CPDLC.AircraftPDU
{         ATCDownlinkMessage
                  {header: ATCMessageHeader
                           MsgIdentificationNumber "{ 1 }"
                           DateTimeGroup "{ 1997/09/03 16:13:26 }"}
                  {elementIds:
                           ATCDownlinkMsgElementId
                                    "{WILCO}" (Null)}
}
```

Hexadecimal view (13 octets)

00 25 22 60 04 6E C0 40 C0 A0 6B 40 00

Binary view

	T-DATA User data
	Fully-encoded-data ::= SEQUENCE SIZE (1, ...) OF PDV-list
0	Extension bit: no extensions, 1 element in SEQUENCE OF
	PDV-list ::= SEQUENCE
0	Bitmap - no optional elements in PDV-list
	Presentation-context-identifier ::= INTEGER (1..127, ...)
0	Extension bit: no extensions, size is constrained to 1 .. 127
0 0000.00	Constrained value = 1 (acse-apdu)
10	choice 2 = arbitrary (BIT STRING) encoding
0101.0010	length determinant = 52H= 82 (dec) bits
	ACSE APDU
0	Extension bit: No extension values present in ACSE-apdu
010.	Indicates RLRQ-apdu is used (third item in ACSE-apdu CHOICE)
0	Extension bit: No extension value present in RLRQ-apdu
11	Bitmap indicates presence of OPTIONAL fields: reason, user-information
	reason : Release-request-reason ::= INTEGER (0|1|30, ...)
0	Extension bit: No extension values present
0000.0	value = 0 (normal)
	user-information
0	Extension bit: No extension values present in user-information, so SEQUENCE OF is exactly 1 in length
00 0	Bitmap indicates no OPTIONAL fields present in EXTERNAL type
10	Choice 2 = BIT STRING encoding
0.0110 111	Length determinant for bit string = 37H = 55 bits (dec)
	CPDLC.AircraftPDU
0.	Extension bit: No extension value present in AircraftPDUs

11	CHOICE = ATCDownlinkMessage
	header: ATCMessageHeader
00	Bitmap: no OPTIONAL or DEFAULT present
0000.01	MsgIdentificationNumber ::= INTEGER (0..63), value = 1
	DateTimeGroup
00 0000.1	Date.Year ::= INTEGER (1996..2095), constrained value = 1997
100 0	Date.Month ::= INTEGER (1..12), value = 09
000.10	Date.Day ::= INTEGER (1..31), value = 03
10 000	Time.TimeHours ::= INTEGER (0..23), value = 16
0.0110 1	Time.TimeMinutes ::= INTEGER (0..59), value = 13
011.010	TimeSeconds ::= INTEGER (0..59), value = 26
	elementIds: SEQUENCE SIZE (1..5) OF
0 00	SIZE = 1
	ATCDownlinkMsgElementId
0	no extensions present in ATCDownlinkMsgElementId
0.0000 00	CHOICE 1 of 114 = NULL (WILCO)
00.	padding

8. FUTURE MIGRATION

8.1 Migration Path of the ATN Upper Layers

8.1.1 This section indicates some of the developments in the ULA that are likely in future CNS/ATM packages. The intention is to assist developers and specification writers to build suitable "hooks" into their implementations and specifications.

8.2 Forward Compatibility Provisions

8.2.1 The ATN ULCS and applications were designed for forward compatibility. The means for forward compatibility include extension markers in all abstract syntaxes and version markers in all application specifications. Care was taken in the ISO efficiency enhancements to ensure that the "fast byte" session and presentation protocols incorporate "escape" mechanisms allowing future additions to be made that are distinguishable at the protocol level. For example, the negotiation of alternative encoding rules could be added in future to the presentation layer. A flexible and extensible naming hierarchy has been defined.

8.2.2 The use of extension markers in abstract syntaxes means that an "extension bit" is encoded at appropriate places into the PER-generated bit-stream. If an extension bit is set to the value zero, then no extensions are present, and a decoder with a knowledge of the basic abstract syntax can fully decode any received vale. If an extension bit is set to the value one, then the decoder knows that extensions have been added to the abstract syntax. If it does not know the nature of these extensions (i.e. it is an older version than the sender), then it can make use of length encodings to skip over the extensions.

8.3 Optimisation of Upper Layer Connection Protocols

8.3.1 The session and presentation layer efficiency enhancement addenda provide support for the "Fast Associate" mechanism, in which an Upper Layer Context Identifier (ULCTXID) can be sent using the T-CONNECT or T-DATA services, and this will allow the semantic content of ACSE, Presentation and Session connect PDUs to be sent in a highly compressed fashion.

8.4 Multiple Associations per Transport Connection

8.4.1 One significant enhancement will be to allow many application associations or ASO-associations to be hosted on a single underlying transport connection. This will enable considerable savings in resources and in the overhead of setting up many transport connections and using many TSAPs.

8.4.2 For example, the AIDC application currently establishes a separate transport connection between ground stations for each flight. Using a shared transport connection would enable a single transport connection between each pair of ground stations with flights treated on multiplexed associations on the transport connection.

8.4.3 Support for this functionality is being built in to the third edition of the ISO/IEC ACSE standards, and it is intended to make use of this standard when it is developed. This will have the added advantage of guaranteed backwards compatibility with the CNS/ATM-1 specifications.

8.4.4 Two mechanisms to allow this multiplexing to take place are included in ACSE edition 3. These are:

a) *Higher level associations*. This allows ASOs to set up ASO-associations with peer ASOs using the facilities of ACSE alone. No session or presentation layer functionality will be available.
b) *Nested connections*. This allows for session and presentation connections to be recursively nested. The full session and presentation functionality is available to ASOs.

8.4.5 The current standardisation work in the ATN Upper Layers focuses on the developments of ACSE, edition 3. The new work on ACSE allows for explicit support of the extended application layer structure (XALS) concepts by ACSE. The work also allows ACSE to support multiple associations over a single TP4 connection.

8.4.6 To assist in application design, a standardised application layer architecture has been developed as ISO/IEC 9545 Application Layer Structure (ALS).

8.4.7 The following figure shows the various components of in the application layer as defined in the Application Layer Structure (ALS) model in ISO/IEC 9545, and shows how they are related.

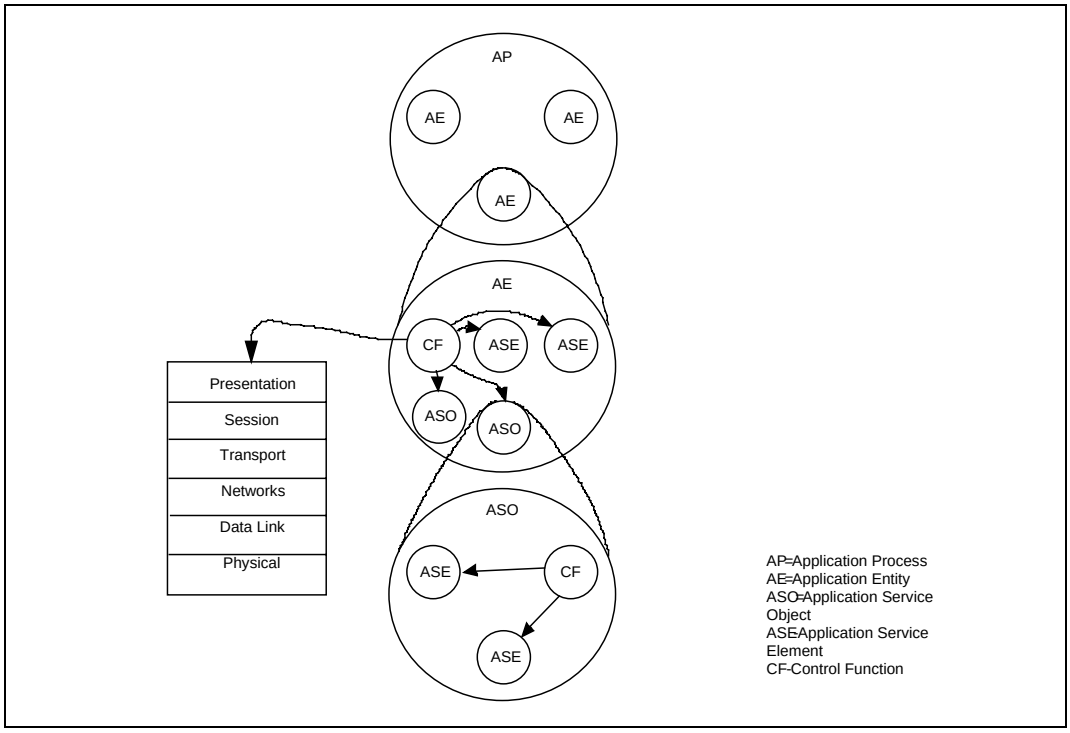


Figure 8-1. Components of the application layer as defined in ALS

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

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

8.4.10 An ASO association is between two (or possibly more) ASOIs. These peer ASOIs are not

necessarily of the same type, but must be of complementary types. For example, if they are to exchange data, both must understand the same data syntax.

8.4.11 ACSE edition 3 standards will 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

8.5 Additional Common Services

8.5.1 For CNS/ATM-1, there will only be two ASEs in the AE; namely ACSE and the ATN-App ASE.

8.5.2 Depending on user requirements, a number of ASEs may be defined in the future 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 would be necessary or even desirable to implement the complete set of selected upper layer profiles on all end-systems. Subsets of the full set could be selected, to provide appropriate levels of functionality to meet the requirements of different classes of applications.

8.5.3 An Application Service Object (ASO) template is under consideration to enable future ASEs and ASOs and their associated CF to be specified in a formalised way. This will also decouple the application user from concerns about managing communication stack connections, by provision of an "implicit start".

8.5.4 Common services are being developed for ATN security (e.g., the use of X.509 in the ATN), and systems management (e.g., profiling of CMIP and Managed Objects for the ATN).

8.6 Connectionless Upper Layer Architecture

8.6.1 A profile for the connectionless upper layer architecture is under development. This entails a connectionless dialogue service offering a unit-data service over the ATN connectionless transport service currently profiled in Sub-Volume 5. The profile enhances ISP 11188-4 :1996 common upper layer requirements profile for ATN use. The profile includes the ISO connectionless ACSE (ISO/IEC 10035-1:1995), connectionless presentation (ISO/IEC 9576-1 :1995), and connectionless session (ISO/IEC 9548-1 :1995) protocols.

9. IMPLEMENTATION DECISIONS

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

9.1.2 A possible implementation approach is illustrated in Figure 9-1, which shows how the finite state machines (FSMs) at each layer are interrelated.

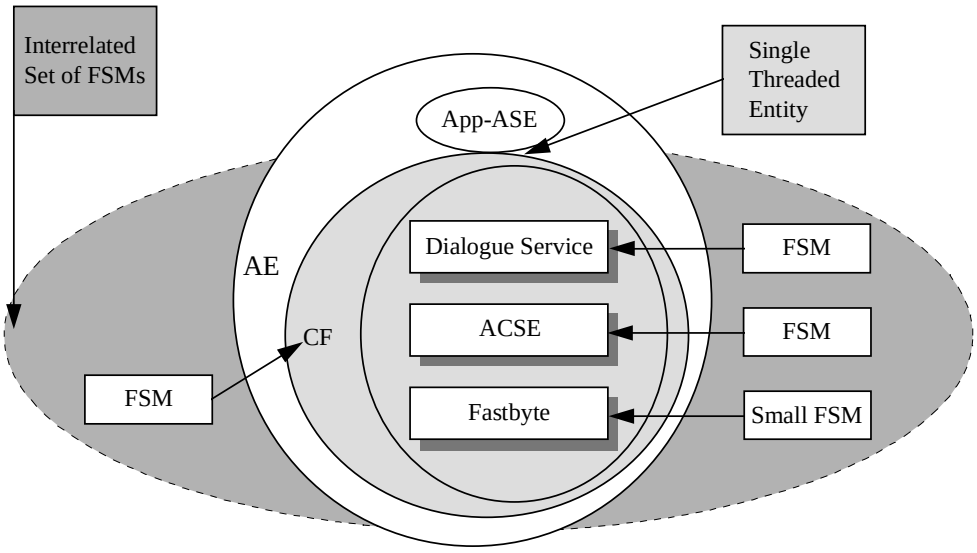


Figure 9-1. Possible Implementation Strategy

9.1.3 The CF in particular is defined in a manner that implies a multi-threaded software design. In tracing the CF and the interactions with ACSE and/or the user, calls are made to the CF state machine in a recursive fashion that requires careful attention to maintaining state information independent of the CF itself.

9.1.4 The order of actions described in individual cells of the state machine is often not the order which may be needed for proper operation of an implementation. Care must be exercised to ensure that state transitions in the CF are taken before transfer of control is made to other module (such as ACSE) to ensure that upon recursive calls to the CF it is in the proper state.

9.1.5 While the ULCS defines the Dialogue Service abstract interface, there is no requirement to actually create such an interface as long as the application/ULCS interface provides the appropriate functionality. The Dialogue Service interface does define a sufficient interface if an implementation chooses that as its application interface.

9.1.6 Retrieval of calling AE qualifier value

9.1.7 If the Calling Peer ID is given in the D-START request primitive (as described in 4.3.3.3.2), then the dialogue service provider has to send the Calling AE qualifier as described in 4.3.2.1 and 4.3.2.3. For some implementations, the dialogue service provider is an "autonomous and independent" service provider without any assumptions about its service users or based on different used SAPs. So with the current service definition, it is unable to retrieve this AE qualifier value.

9.1.8 It might be thought therefore that the AE qualifier should be present in the list of parameters of the D-START service. In fact, this is an implementation issue. The way that the Calling AP Title and the Calling AE Qualifier are retrieved is a local implementation matter.

10. ANNEX A - Naming, Addressing and Registration

Note.— This section was formerly Section 3.1 of the UL Guidance Material. It is proposed to remove this material from this document and include it instead in Part II of the Comprehensive ATN Manual (CAMAL).

10.1.1 This section contains material on upper layer naming and addressing.

10.1.2 "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."*

10.1.3 Naming

10.1.4 Information objects which may need to be named include:

- application processes
- managed objects
- ATN application message types

10.1.5 AP, AE and Context Naming

10.1.6 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.

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

10.1.8 AP-titles and AE-qualifiers may be assigned either an attribute-based name form (Directory Name) 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. For CNS/ATM-1, only the Object Identifier name form is used.

10.1.9 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. The use of invocation identifiers is not required for CNS/ATM-1.

10.1.10 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.

10.1.11 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 only within the scope of the co-operating AE-invocations, and thus do not have to be registered.

10.1.12 In connection-mode communications, the AET can be used in called, calling and responding application address parameters in A-ASSOCIATE 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.

10.1.13 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, and is further defined for ATN in the ULCS SARPs.

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

10.1.15 The application context name is a simple construct, since the CNS/ATM-1 ASE list can be inferred from the AE-title. The CNS/ATM-1 application context name is used only to distinguish between different versions of an application context within the scope of a given AE type, as identified by the AE qualifier.

10.1.16 Addressing

10.1.17 According to ISO/IEC 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 Presentation selector are not utilised in the CNS/ATM-1 Package.

10.1.18 For CNS/ATM-1, there is no upper-layer addressing. All addressing of the AE-type is complete with the TSAP address. All upper layer selectors are necessarily absent.

10.1.19 The only mandatory addressing parameters in the A-ASSOCIATE service primitives are the calling, called and responding Presentation Addresses. These are passed transparently by ACSE to the Presentation Service. As there are no presentation or session selectors, the PSAP address will be identical to the TSAP address, as follows, where the terms "Long TSAP" and "Short TSAP" are as used in the CM SARPs.

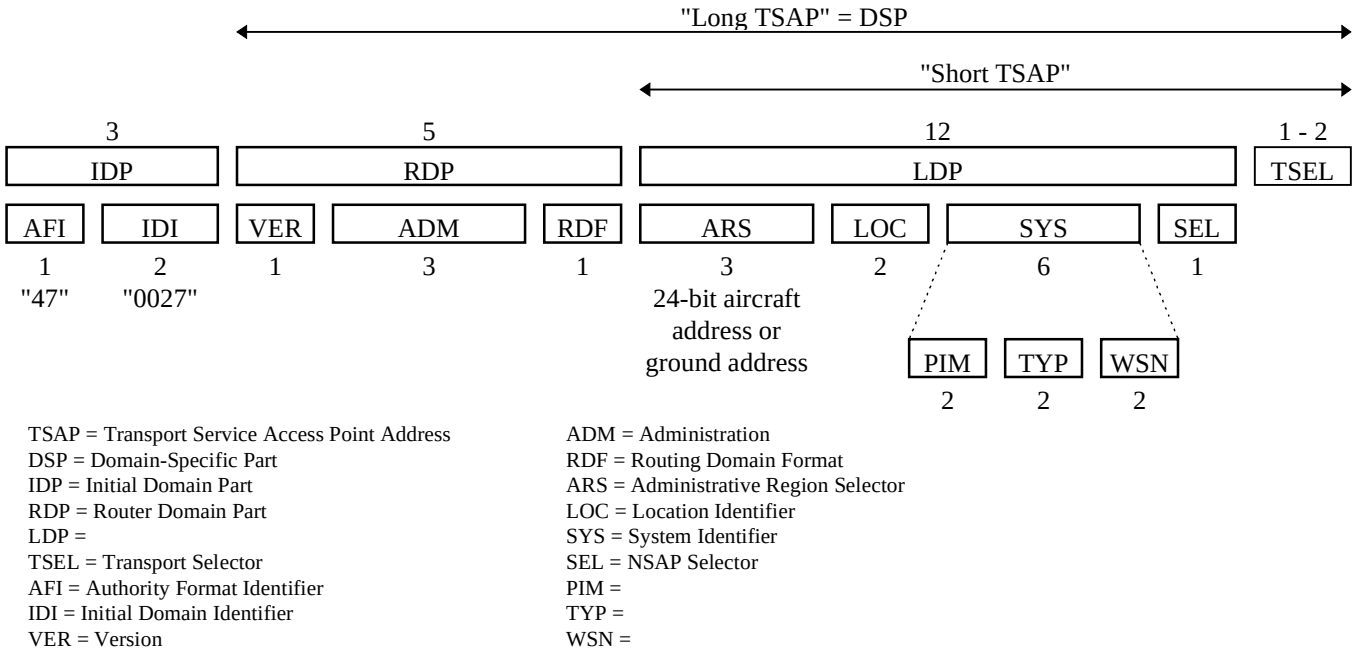


Figure 10-1. Composition of TSAP Address (= PSAP Address)

10.1.20 Name - Address Mapping

10.1.21 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 associated with one or more PSAPs 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. The Application Layer directory function provides a mapping from an AET into the PSAP address required to access the referenced application entity.

10.1.22 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.

10.1.23 In CNS/ATM-1, the type of the AE is the same as the type of the ATN-App ASE. That is for example, an ADS AE will contain only an ADS ASE and ACSE. Thus, on connection establishment, the peer title can be completely constructed: the variable part of the AP-title (the peer end system ID) is provided by the "Called/Calling Peer ID" value in the dialogue service, and the peer AE-qualifier (e.g., CPC) is the same as the receiving application's.

10.1.24 Registration Issues

10.1.25 Under the ISO/IEC 9834 registration scheme, ICAO has been assigned an International Code Designator (ICD) of "icao (27)"..

10.1.26 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. Ideally, a given

object is only assigned one OID, i.e. registered only once (either by ICAO or by some other organisation).

10.1.27 ICAO Working Groups can register information objects including ASOs, ASEs, Application Titles, Presentation Contexts. It may also be necessary to set up a registration authority for Distinguished Names, as used by the Directory service and by systems management.

10.1.28 Names, in the form of object identifiers (OIDs), are assigned to defined ATN entities in the ULCS SARPs, i.e., the ULCS SARPs is the register for these identifiers.

10.1.29 ISO/IEC 8824-1 specifies the top of the hierarchical OID name space. At the first level, provision is made for ISO, International Telecommunication Union Telecommunication Standardisation Sector (ITU-T) and joint ISO/ITU-T sub-name spaces. The ISO name space is further subdivided into:

- standard (0)
- registration-authority (1)
- member-body (2)
- identified-organisation (3)

10.1.30 ICAO has requested and obtained the allocation of an International Code Designator (ICD), according to ISO 6523. The ICD obtained, name and number “icao (27)”, uniquely identifies ICAO and allows ICAO to establish its own object identifier name space within the International Organisation arc using the prefix: { iso (1) identified-organisation (3) icao (27) }. Similarly, IATA has obtained an ICD of “iata (19)”; values assigned under the IATA name space are out of scope.

10.1.31 Within the ICAO name space, the initial allocation of object identifiers follows the structure and values defined in the ULCS SARPs.

10.1.32 In the future, it is likely that the ATN object identifier tree will have further levels of structure, and that fully location-independent values will be assigned.

10.1.33 The ATN naming hierarchy is illustrated below.

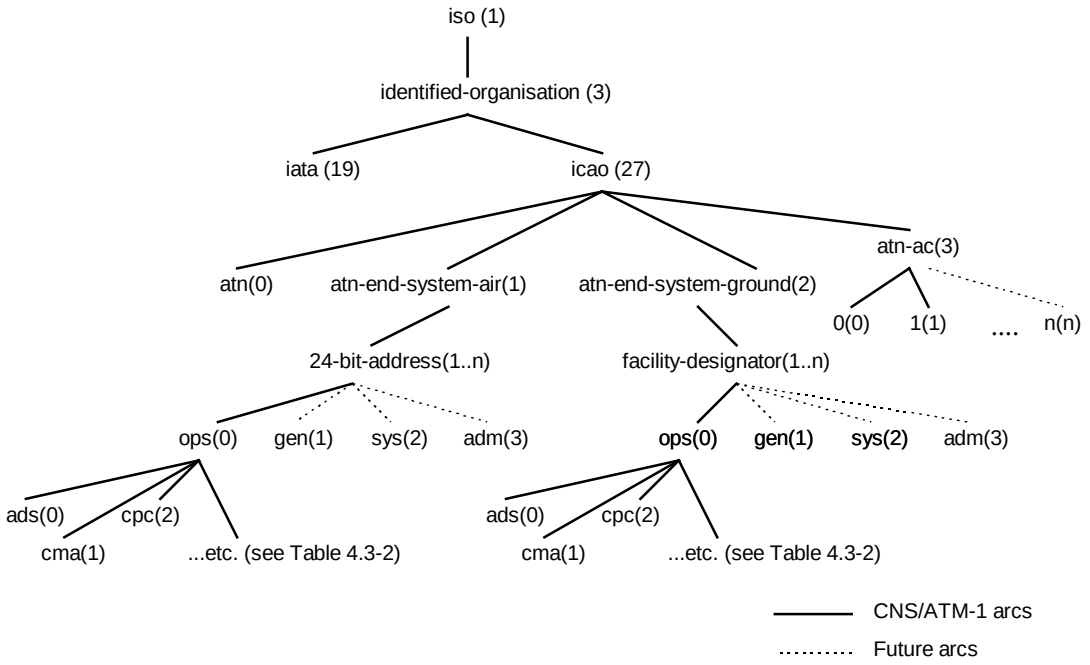


Figure 10-2. ATN Naming Hierarchy

10.1.34 Immediately under the ICAO arc, the values specified in the following table are used to specify the next level of the naming hierarchy.

Table 10-1. Top-level ICAO Identifiers

Name and numeric value	Description
atn (0)	General ATN identifiers
atn-end-system-air (1)	ATN aircraft end systems. The following OID component beneath this arc is a 24-bit ICAO aircraft identifier
atn-end-system-ground (2)	ATN ground end systems. The following OID component beneath this arc is an ICAO facility designator
atn-ac (3)	ATN application context names

10.1.35 Definition of Object Identifiers

Table 10-2. Definition of Object Identifiers

Entity	**Object Identifier**
Application context names	{ iso (1) identified-organisation (3) icao (27) atn-ac (3) version-<v> (v) }
AP-titles	{ iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) <end-system-id> (n) operational (0) } or: { iso (1) identified-organisation (3) icao (27) atn-end-system-ground (2) <end-system-id> (n) operational (0) }

AE-titles	{ iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) <end-system-id> (n) operational (0) <ae-qualifier> (m) } or { iso (1) identified-organisation (3) icao (27) atn-end-system-ground (2) <end-system-id> (n) operational (0) <ae-qualifier> (m) }
ATN Security labels	{ iso (1) identified-organisation (3) icao (27) atn (0) traffic-type-and-routing-policy (0) }

where:

v is an INTEGER in the range 0..255. (The value n = 0 is reserved for use by the CF).

<end-system-id> is the ICAO 24-bit address for aircraft end systems, or the ICAO facility designator for ground end systems.

(n) is an INTEGER value derived from the <end-system-id>, using the algorithm defined in the ULCS SARPs

<ae-qualifier> is the name form of the AE qualifier from Table 4.3-2 of the ULCS SARPs.

(m) is the number form of the AE qualifier from Table 4.3-2. of the ULCS SARPs

10.1.36 Examples of OID values and their encodings are given in the following table.

Table 10-3. Examples of Object Identifier Encodings

Entity	**Object Identifier**	**BER / PER Encoding (Hex)**
Application context name for version 1 applications	{ iso (1) identified-organisation (3) icao (27) atn-ac (3) version-1 (1) }	2B 1B 03 01
AP-title for ADS-air	{ iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) 000000011011011001100110 (112,230) operational (0) }	2B 1B 01 86 EC 66 00
AP-title for CM-ground	{ iso (1) identified-organisation (3) icao (27) atn-end-system-ground (2) LFPODLHX (n) operational (0) }	2B 1B 02 8C 86 90 8F 84 8C 88 18 00
AE-title for ADS-air	{ iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) 000000011011011001100110 (112,230) operational (0) ADS (0) }	2B 1B 01 86 EC 66 00 00
AE-title for CM ground	{ iso (1) identified-organisation (3) icao (27) atn-end-system-ground (2) LFPODLHX (n) operational (0) CMA (1) }	2B 1B 02 8C 86 90 8F 84 8C 88 18 00 01
ATN Security Registration ID	{ iso (1) identified-organisation (3) icao (27) atn (0) traffic-type-and-routing-policy (0) }	2B 1B 00 00

Note 1.— The 24-bit aircraft identifier in this example {000000011011011001100110} equates to a decimal value of 112,230, or hexadecimal {01 B6 66}. This encodes as hex {86 EC 66} using the rules for encoding of object identifier sub-identifiers in ISO/IEC 8825-1. That is, the top bit of each octet indicates whether this is the last octet in the series, and the remaining 7 bits of each octet are concatenated to form the encoded value. (See 5.4 for further explanation)

Note 2.— The ground station identifier in this example {LFPODLHX} equates to an extremely large decimal value which is not evaluated in this example. This encodes as hex { 8C 86 90 8F 84 8C 88 18} using the encoding rules specified in 4.3.2.4.2 of the ULCS SARPs. That is, the top bit of each octet indicates whether this is the last octet in the series, the next bit is always zero, and the remaining 6 bits of each octet encode one character, where A = 1, B = 2, etc. This gives compatibility with the standard OID encoding. (See 5.4).