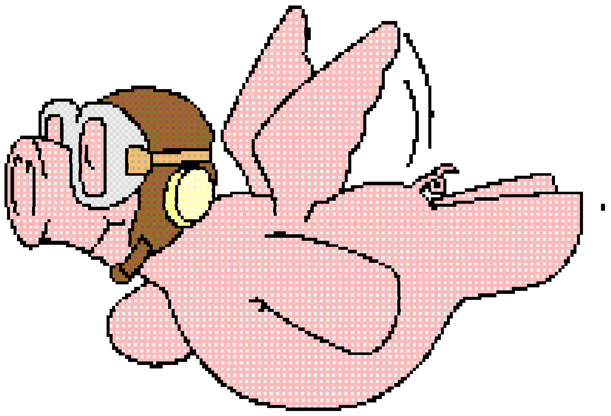
The ATN SARPs



Subvolume Four

Upper Layer Communications Service (ULCS)

Thrid Edition

(Final Editor's Draft)

This PDF version has been prepared for the ATNP Working Groups by Helios Information Services Ltd. – http://www.helios-is.com

Errata and Disclaimer

Please note that this document has been prepared from a number of separate files and no attempt has been made to ensure continuity of page numbers. You may therefore find some overlap between page numbers.

This document has been prepared on a “best efforts” basis and no warrantee is offered as to its correctness.

FOREWORD

The material contained in this document was originally developed as the detailed part of the first set of Standards and Recommended Practices (SARPs) for the aeronautical telecommunication network (ATN) which has commonly been referred to as the CNS/ATM-1 Package. It was intended to make the material an appendix to the new Chapter 3 of Annex 10, Volume III, Part I, containing broad, general, stable and mostly regulatory-type provisions (the core part of new ATN SARPs).

In December 1997, the Air Navigation Commission (ANC), while conducting the final review of draft ATN SARPs, agreed that the detailed part of ATN SARPs should be published as an ICAO manual (to be updated annually, if necessary), while retaining its SARPs-style language. The ANC has reviewed the status of the document in light of continuing worldwide ATN implementation. The Third Edition includes amendments from implementors and regulatory authorities, as well as four new Sub-Volumes to answer requirements for further standardization, in the interests of safety, regularity and efficiency of international civil aviation.

This document consists of nine Sub-Volumes:

Sub-Volume I	— Introduction and System Level Requirements
Sub-Volume II	— Air-Ground Applications
Sub-Volume III	— Ground-Ground Applications
Sub-Volume IV	— Upper Layer Communications Service (ULCS)
Sub-Volume V	— Internet Communications Service (ICS)
Sub-Volume VI	— System Management (SM)
Sub-Volume VII	— Directory Services (DIR)
Sub-Volume VIII	— Security (SEC)
Sub-Volume IX	— Registration (REG)

Provisions contained in Sub-Volumes II, III, IV, V, VI, VII, VIII, and IX have been developed in accordance with system requirements specified in Sub-Volume I.

In line with the agreement by the ANC that the document should be updated on a yearly basis (if deemed necessary), the Third Edition has been published to incorporate changes necessitated by continuing validation and actual implementation activities.

TABLE OF CONTENTS

SUB-VOLUME V. INTERNET COMMUNICATIONS SERVICE

4.1 INTRODUCTION

4.1.1 Scope and Objectives

4.1.1.1 The ATN Upper Layer (UL) communications service is specified here.

4.1.1.2 The UL specification supports all current ATN applications except the ATS Message Application. This specification is designed to optimise the use of communications bandwidth, and consequently restricts the functionality available from the OSI Session and Presentation layers.

4.1.1.3 The ATN requirements are addressed for Session Layer (Layer 5), Presentation Layer (Layer 6), and a part of the Application Layer (Layer 7) of the OSI reference model. Figure 4.1-1 shows a conceptual view of the scope of the UL communications service. The remaining part of the Application Layer is the province of the individual ATN applications (i.e. the ADS, CM, CPDLC and FIS (ATIS) specifications for air-ground applications, and the ICC (AIDC) specifications for ground-ground applications).

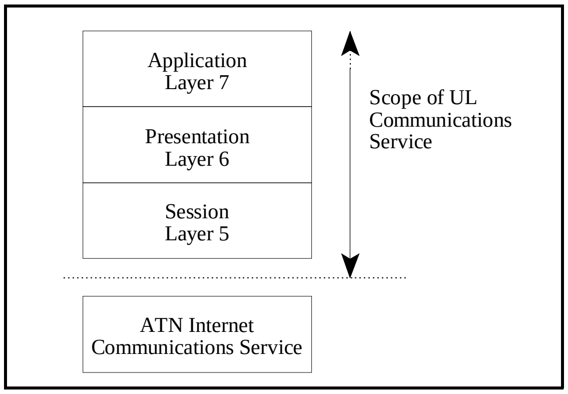

Application
Layer 7
Presentation
Layer 6
Session
Layer 5
Scope of UL Communications Service
ATN Internet Communications Service

Figure 4.1-1. Conceptual view of the scope of the UL Communications Service

4.1.2 Background

4.1.2.1 The communication aspects of the ATN applications are modelled as Application Entities (AEs) (see 4.1.4.2). Figure 4.1-2 illustrates an example of the application layer structure for the ATN applications.

4.1.2.2 The specification of the UL communications service includes a profile for the protocols in the upper layers, an AE structure and a number of common application services.

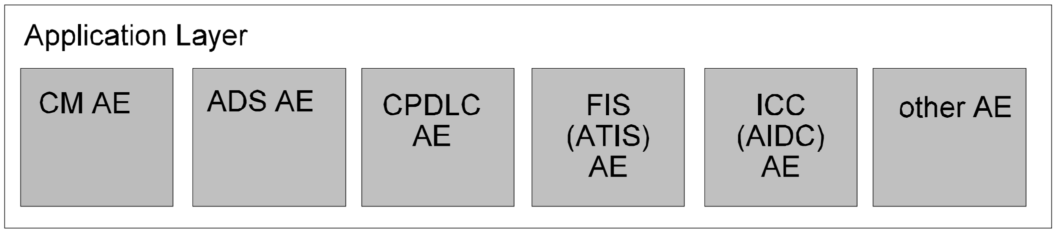


Figure 4.1-2. Conceptual view of Application Layer

4.1.3 Structure of UL Communications Service Specification

4.1.3.1 This specification is structured as follows:

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 connection-oriented 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).

g) Connectionless Dialogue Service and Profile (4.7) describes the abstract service which may be used to provide applications with a datagram (D-UNIT-DATA) service, and specifies the requirements for connectionless CF, Session, Presentation and ACSE protocols.

h) Security Application Service Object Description (4.8) describes the Security ASO, which may be used to provide a secured dialogue service.

i) Generic ATN Communication Service (GACS) Specification (4.9) describes the GACS ASE, which builds on the dialogue service specified in 4.2 and 4.7 to provide either an enhanced dialogue service to other ASEs, or an AE which provides a simple mechanism for transferring user data across the ATN, with or without end-to-end confirmation.

4.1.4 Upper Layer Functionality

4.1.4.1 Upper Layer Profile Overview

4.1.4.1.1 A profile is specified for the connection-oriented protocols of Session layer, Presentation layer and the Association Control Service Element (ACSE).

4.1.4.1.2 The Session portion of the specified profile is based on the efficiency enhancements to the Session protocol which are standardised in ISO/IEC 8327-1: 1996 / Amd. 1: 1997 | ITU-T Rec. X.225 (1995)/Amd.1 (1997).

4.1.4.1.3 The Presentation portion of the specified profile is based on the efficiency enhancements to the Presentation protocol which are standardised in ISO/IEC 8823-1: 1994 / Amd. 1: 1997 | ITU-T Rec. X.226 (1994)/Amd.1 (1997).

4.1.4.1.4 As a consequence of using the Session and Presentation protocol efficiency enhancements, the protocol control information transferred by these protocols amounts to two octets in each direction during the connection establishment phase, and zero octets at all other times.

4.1.4.1.5 The ACSE portion of the specified profile is based on ISO/IEC 8650-1 | ITU-T Rec. X.227 (1994), including the extensibility notation as specified as Amendment 1 to that standard.

4.1.4.2 Application Entity (AE) Structure

4.1.4.2.1 The specified AE structure is based on the application layer structure defined in ISO/IEC 9545 | ITU-T Rec. X.207 (1993), where the concepts of Application Service Element (ASE), Application Service Object (ASO) and Control Function (CF) are defined.

4.1.4.2.2 Figure 4.1-3 shows the generic structure of an AE with arrows representing the abstract service boundaries of the various elements. The “upper” service boundary is the abstract service provided by an ASE to its user(s). The “lower” service boundary is the abstract service which is provided to the ASE by the CF.

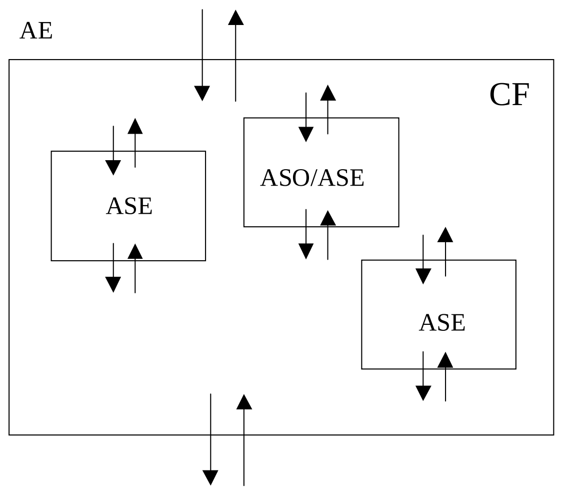


Figure 4.1-3. Generic Application Entity structure

4.1.4.2.3 The ASE is an element engineered to perform a required task. ISO/IEC 9545 | ITU-T Rec. X.207 describes how two or more ASEs may be combined, together with a CF to co-ordinate their operation to form an ASO. In turn, an ASO may be combined with other ASOs or ASEs with a CF to form larger ASOs. The AE is the outermost ASO.

4.1.4.3 Application Services

4.1.4.3.1 For each of the current ATN applications a specific ASE exists, and is defined in the relevant ATN Application specification. The generic name "ATN-App ASE" is used for these specific ASEs.

4.1.4.3.2 Various abstract services are specified. The services are provided at abstract service boundaries. The abstract service provided by the AE to the Application-user (i.e. the service provided at the upper boundary of the AE) is specified in 4.3. In the AE structure specified here, this service is a pass-through to the ATN-App ASE.

4.1.4.3.3 Figure 4.1-4 shows the AE structure which is used to model the ATN applications. This is described in detail in 4.3.

4.1.4.3.4 The Dialogue Service (DS) as defined in 4.2 is the abstract service which the ATN-App ASEs use to interact with the UL communications service. That is, the DS is the combination of specific internal primitives made available by the CF at the lower boundary of the ATN ASE/ASO - it is the application's "world view". In order to provide this service, the CF uses the services of ACSE.

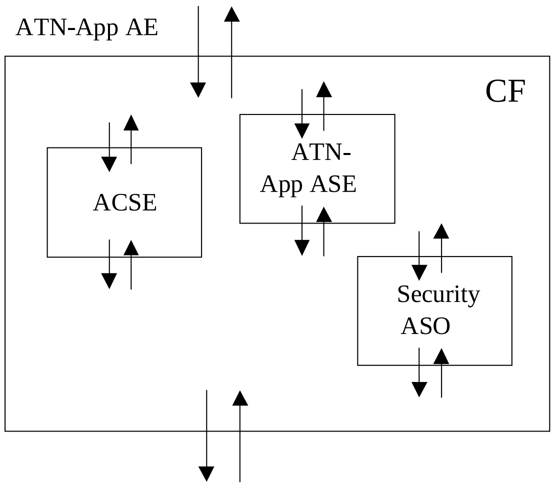


Figure 4.1-4. ATN-specific AE structure

4.1.5 Conventions

4.1.5.1 In the tables of service primitives throughout this UL Communications Service specification, the presence of each parameter is described by one of the following values:

a) blank not present;

b) C conditional upon some predicate explained in the text;

c) C(=) conditional upon the value of the parameter to the immediate left being present, and equal to that value;

d) M mandatory;

e) M(=) mandatory, and equal to the value of the parameter to the immediate left;

f) U user option.

4.1.5.2 The following abbreviations are used in the various service descriptions and protocol tables:

a) Reqrequest; an invocation of a service primitive initiated from the user of an abstract service and submitted to the service for action;

b) Ind indication; an invocation of a service primitive delivered from the abstract service to a user of the service;

c) Rspresponse; an invocation of a service primitive submitted by the user of an abstract service in response to a previous indication, in the case of a confirmed service.

d) Cnf confirmation; an invocation of a service primitive delivered from the abstract service to a user of the service, which confirms that a previous request primitive from that user has been acted upon by the service, in the case of a confirmed service.

4.1.5.3 An unconfirmed service allows a single data transfer, in one direction, without the semantics of a response in the opposite direction..

4.1.5.4 A confirmed service provides confirmation to a service user of the outcome of an invocation of that service, for example that a data transfer initiated by that user was delivered to its peer user.

4.1.5.5 An abstract service is a syntactical description of a parameter which does not imply a specific implementation.

4.2 DIALOGUE SERVICE DESCRIPTION

4.2.1 Scope of Dialogue Service

4.2.1.1 Implementations of the ATN-App ASE, together with the UL elements which provide the Dialogue Service (DS), shall exhibit the behaviour defined in this abstract service definition.

Note 1.— The Dialogue Service is the abstract service which is used by an ATN-App ASE at its lower service boundary. There is no requirement to implement the DS in any product. ATN end systems will in general be designed in such a way that it is impossible to detect (from external access) whether or not an interface corresponding to the DS has been built.

Note 2.— The DS is described from the viewpoint of the ATN-App ASE, using abstract service definition conventions. The abstract service definition is a descriptive technique used to specify the behaviour exhibited by part of the ATN application layer. Specifications of application service elements (ASEs), such as the specifications of ADS, CPDLC, CM and ATIS, may include common functionality by reference to the DS. The DS allows ATN-App ASEs to be specified without the need to consider some of the complexities of some aspects of the underlying communications.

Note 3.— The DS supports a communication relationship between two peers for a duration which exists until the peers agree to terminate the relationship or the relationship is aborted.

Note 4.— The DS defines a service which may be used to support an ATN-App ASE at its "lower" service boundary. Such an ASE is denoted a ***DS-User****. The DS-User can be specified to use the DS in a variety of ways that can be defined in terms of reliability characteristics. A number of user-visible service levels can thus be offered, including for example the following:*

a) An unconfirmed service, which allows individual messages to be transmitted after a dialogue has been set up.

b) A confirmed service, which provides end-to-end confirmation that a message sent by one DS-User was received and acknowledged by the peer DS-User.

Note 5.— An implementation of the DS provider will typically be responsible for detection of errors such as:

a) Invalid primitive (primitive unknown or error in parameter(s))

b) Invalid sequence (primitive issued at inappropriate time)

c) Insufficient resources on submission

d) Invalid or unreachable recipient on submission

e) Data field too large on receive (local implementation constraint exceeded)

f) Invalid or unreachable recipient on receive.

Note 6.— An implementation of an ATN application which makes use of the DS has to be designed with error handling procedures for local error conditions.

4.2.2 Service Primitives

4.2.2.1 Implementations which claim to support the DS functionality shall exhibit the behaviour defined by the service primitives in Table 4.2-1.

Table 4.2-1. Summary of Dialogue Service primitives

Service	**Description**
D-START	This is a confirmed service used to establish the binding between the communicating DS-Users.
D-DATA	This unconfirmed service is used by a DS-User to send a message from that DS-User to the peer DS-User.
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.
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.
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.
D-UNIT-DATA	This unconfirmed service is used to send a single data item from one peer DS-User to another. Any problem in delivering the data item to the recipient will not be signalled to the originator. This service is specified in 4.7.

Note.— Table 4.2-2 lists the parameters used when invoking the services.

Table 4.2-2. Parameters of the Dialogue Service primitives

Service	**Parameters**
D-START	Called Peer ID Called Sys-ID Called Presentation Address Calling Peer ID Calling Sys-ID Calling Presentation Address DS-User Version Number Security Requirements Quality-of-Service Result Reject Source User Data
D-DATA	User Data
D-END	Result User Data
D-ABORT	Originator User Data
D-P-ABORT	(no parameters)

4.2.3 Service Definition

4.2.3.1 Sequence of Primitives

4.2.3.1.1 Implementations which claim to support the DS functionality shall exhibit behaviour allowing two communicating DS-Users to:

a) establish a dialogue;

b) exchange user data;

c) terminate a dialogue in an orderly or abnormal fashion; and

d) be informed of DS abnormal dialogue termination due to the underlying communications failure;

consistent with the appropriate use of the corresponding service primitives.

Note 1.— Either DS-User may send data at any time after the initial D-START exchange, by using the D-DATA service. Under normal circumstances, a dialogue is released by a DS-User invoking the D-END service. A dialogue is abnormally released with the D-ABORT service. If the underlying service provider abnormally releases the dialogue, the DS-Users which are aware of the dialogue will be notified with the D-P-ABORT service.

Note 2.— For the purposes of this service definition, it is only valid for the DS-User to issue and be prepared to receive primitives for one Dialogue according to the permitted sequences of DS primitives shown in Table 4.2-3, where intersections marked "Y" show possible primitives which may occur after the primitive in the column heading.

Table 4.2-3. Sequence of DS primitives for one Dialogue at one DS-User

The DS primitive X -> may be followed by the DS primitive Y	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***
1 D-START req													
2 D-START cnf (accepted)	*Y*												
3 D-START ind								*Y*		*Y*	*Y*	*Y*	*Y*
4 D-START rsp (accepted)			*Y*										
5 D-DATA req		*Y*		*Y*	*Y*	*Y*			*Y*				
6 D-DATA ind		*Y*		*Y*	*Y*	*Y*	*Y*						
7 D-END req		*Y*		*Y*	*Y*	*Y*							
8 D-END cnf (accepted)							*Y*						
9 D-END ind		*Y*		*Y*	*Y*	*Y*							
10 D-END rsp (accepted)									*Y*				
11 D-ABORT req	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*		*Y*				

The DS primitive X -> ***may be followed by the DS primitive Y***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***
12 D-ABORT ind	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*		*Y*				
13 D-P-ABORT ind	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*		*Y*				

Note 3.— For compactness, each DS primitive is given a number in the column headings in Table 4.2-3; the numbers have the meanings assigned in the row headings. For simplicity, where D-START and D-END response and confirmation primitives are used, Table 4.2-3 only shows the case where the D-START or D-END request is accepted by the peer. If a D-START request is rejected, then a DS-User may not issue or receive any other primitives apart from D-START request or indication. If a D-END request is rejected, then a DS-User may continue to issue and receive primitives as if the dialogue had just been established. A D-START request results in a new instance of communication with the peer DS-User, so could occur at any time. Table 4.2-3 only applies to a single instance of communication.

4.2.3.2 The D-START service

4.2.3.2.1 The behaviour defined by the D-START service primitive shall be provided to enable the setting up of a dialogue between two DS-Users.

Note 1.— D-START is a confirmed service which is invoked by a DS-User (the dialogue-initiator) to start a dialogue with a peer DS-User. D-START request, indication, response and confirmation primitives are defined, as illustrated in Figure 4.2-1.

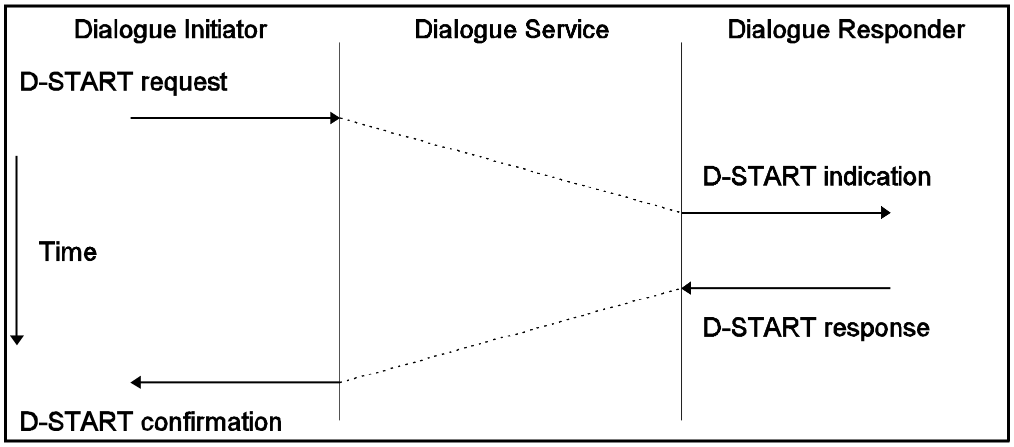


Figure 4.2-1. D-START sequence diagram

Note 2.— The initiating DS-User issues a D-START request primitive. It is not then valid to issue any other primitives (except D-ABORT) until a D-START confirmation is received. When the responding DS-User receives the D-START indication primitive, it must decide whether or not to accept this instantiation of the dialogue service. It may issue only a D-START response or a D-ABORT request primitive. The D-START response and confirmation primitives contain a Result parameter which defines whether the responding DS-User accepts or rejects the request. If the responding DS-User accepts the request, then the dialogue is established. If it rejects the request, then no dialogue exists. The parameters of the D-START primitives are specified in Table 4.2-4.

Table 4.2-4. D-START parameters

Parameter Name	*Req*	*Ind*	*Rsp*	*Cnf*
Called Peer ID	*U*			
Called Sys-ID	*C*			
Called Presentation Address	*U*			
Calling Peer ID	*U*	*C(=)*		
Calling Sys-ID	*C*	*C(=)*		
Calling Presentation Address	*U*	*C(=)*		
DS-User Version Number	*U*	*C(=)*	*U*	*C(=)*
Security Requirements	*U*	*M(=)*	*U*	*M(=)*
Quality Of Service	*M*	*M(=)*	*U*	*M(=)*
Result			*M*	*M*
Reject Source				*C*
User Data	*U*	*C(=)*	*U*	*C(=)*

Note 3.— Called Peer identification. The DS-User identifies the intended peer DS-User by specifying either a name or an address in the D-START request primitive. The DS-User therefore specifies a value for one, and only one, of the Called Peer ID and Called Presentation Address parameters in the D-START request. The Called Peer ID parameter is optionally used in the D-START service to specify the name of the intended peer DS-User, and takes an abstract value corresponding to either a 24-bit ICAO aircraft-id or an ICAO facility designator. The Called Presentation Address parameter is optionally used in the D-START service to specify the address of the intended peer DS-User, and takes the value of an ATN PSAP address (equivalent to an ATN TSAP address as defined in 5.4.2, since Session and Presentation selectors are NULL). If the Called Peer ID parameter is used, then the DS-User may additionally specify the Called Sys-ID parameter to explicitly refer to a specific instance of the peer DS-User at the remote location. If the Called Sys-ID parameter is absent, then ANY instance of the DS-User process at the identified location is being addressed. The syntax of the Called Sys-ID is an 8-octet identifier corresponding to the LOC + SYS fields of the ATN TSAP address, as defined in 5.4.3.8.

Note 4.— Calling Peer identification. The DS-User may optionally request that the name or address of the initiating DS-User be conveyed to the peer DS-User in the D-START service. The DS-User therefore specifies a value for neither or one of the Calling Peer ID and Calling Presentation Address parameters in the D-START request. If the Calling Peer ID parameter is used, then the DS-User may additionally specify the Calling Sys-ID parameter to explicitly refer to a specific instance of the local DS-User. If the Calling Sys-ID parameter is absent, then ANY instance of the DS-User process at the local location is being referenced. The presence of each of Calling Peer ID and Calling Sys-ID parameters in the D-START indication primitive is conditional upon them being specified by the DS-User in the D-START request primitive. If the Calling Peer ID is not specified in the D-START request, then the Calling Presentation Address will be present in the D-START indication, regardless of whether it was specified in the D-START *request. The syntax of the Calling Peer ID, Calling Sys-ID and Calling Presentation Address parameters is identical to the corresponding Called parameters described above.*

Note 5.— The DS-User version number allows peer DS-Users to exchange version information. The parameter is optional in the request and response primitives. Its presence in the indication primitive is conditional upon it being specified by the DS-User in the request primitive, and its presence in the confirmation primitive is conditional upon it being specified by the DS-User in the response primitive. If present, it may take any abstract value in the range 1 to 255.

Note 6.— The Security Requirements parameter allows peer DS-Users to agree the type of secured dialogue. Valid abstract values are specified in 8.3-1. The parameter is optional in the request and response primitives, and its omission by the DS-User is equivalent to the abstract value "No Security". If security services are to be used, the abstract value in the response primitive must be the same as the value provided in the D-START indication. The value in the response primitive may be omitted or set to the abstract value "No Security" if the responder is unable or unwilling to establish a secure dialogue.

Note 7.— The Quality Of Service parameter allows the initiating DS-User to specify in the request primitive its requirements for the quality of service (QOS) to be provided for the dialogue. For ATN, the parameter is not modified by the DS-provider, so the value in the indication primitive is equal to the value in the request. The QOS parameter in the response primitive is assumed by the CF to be equal to the value in the indication primitive. The value of the QOS parameter in the confirmation primitive is equal to that present or assumed in the response primitive. The following QOS parameters may be specified:

e) *Routing Class - valid values are defined in Table 5.6-1*

f) *Priority - valid values are defined in Table 1-2*

g) *Residual Error Rate (RER) - valid values are defined in 5.5.2.3.3. (Not significant for ATSC applications).*

Note 8.— If the Routing Class parameter is not provided by the DS-User in the D-START Request primitive, and the DS-User is an ATS application as specified in 2.1 - 2.4, then the default value "ATSC: No Traffic Type Policy Preference" is assumed. If the DS-User is not an ATS application as specified in 2.1 - 2.4, then the default traffic type "General Communications" is assumed.

Note 9.— If a Priority value is not provided by the DS-User in the D-START Request primitive, then the default value "network/systems administration" is assumed.

Note 10.— For the RER parameter, a low error rate corresponds to a high quality connection, and a higher error rate corresponds to a lower quality connection. For ATSC applications, the highest available integrity level is always selected. Other types of application may select the required integrity level in the D-START request, e.g. for compatibility with basic ISO | ITU-T Transport Service providers which do not support the ATN enhanced transport checksum.

Note 11.— The Result parameter specifies whether the requested dialogue start has been accepted. It can take one of the abstract values:

a) *accepted;*

b) rejected (transient); or

c) rejected (permanent).

Note 12.— The Reject Source parameter is present if the Result parameter has one of the values "rejected (transient)" or "rejected (permanent)". It specifies who rejected the start of the dialogue, and can have one of the abstract values:

a) DS user; or

b) DS provider.

Note 13.— The User Data parameter allows the peer DS-Users to exchange data during the D-START service invocation. Its presence in the indication primitive is conditional upon it being specified by the DS-User in the request primitive, and its presence in the confirmation primitive is conditional upon it being specified by the DS-User in the response primitive.

4.2.3.3 The D-DATA service

4.2.3.3.1 The behaviour defined by the D-DATA service primitive shall be provided to enable the exchange of information between two DS-Users.

Note 1.— D-DATA is an unconfirmed service which provides data transfer between peer DS-Users. The D-START service must first have been successfully completed to establish the communication relationship between the peers. Request and indication primitives are defined, as illustrated in Figure 4.2-2.

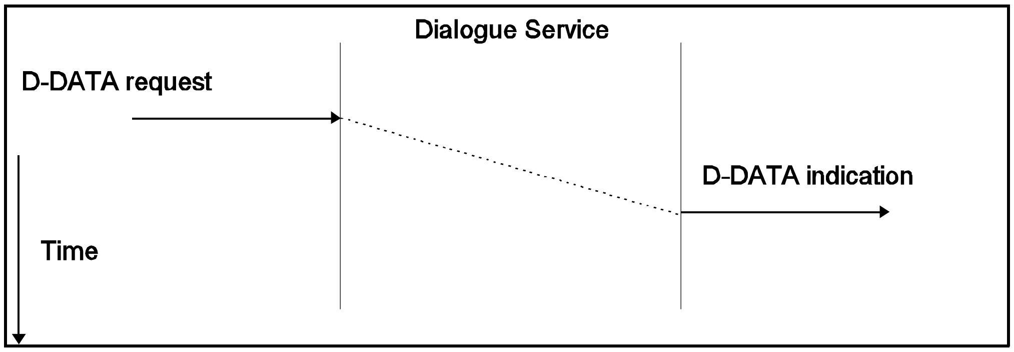


Figure 4.2-2. D-DATA sequence diagram

Note 2.— The parameters of the D-DATA primitives are specified in Table 4.2-5.

Table 4.2-5. D-DATA parameters

Parameter Name	*Req*	*Ind*
User Data	*M*	*M(=)*

Note 3.— The User Data parameter contains the data to be transferred from a DS-User to its peer, using an existing dialogue.

4.2.3.4 **The D-END service**

4.2.3.4.1 The behaviour defined by the D-END service primitive shall be provided to enable the orderly termination of a dialogue between two DS-Users.

Note 1.— D-END is a confirmed service which causes the end of a dialogue. It may be invoked by either of the communicating partners. When the D-END service is invoked, the DS performs an orderly release, whereby any service previously invoked is completed before the dialogue is terminated. The D-END service defines request, indication, response and confirmation primitives, as illustrated in Figure 4.2-3.

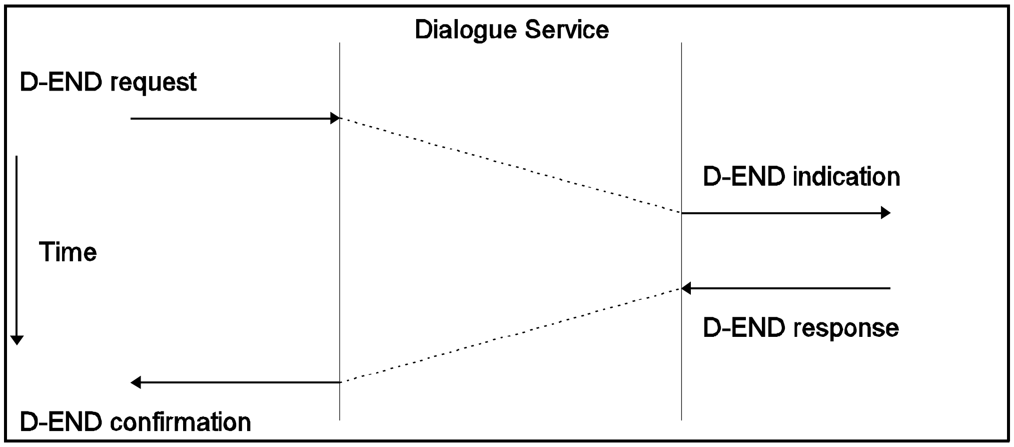


Figure 4.2-3. D-END sequence diagram

Note 2.— The DS-User which wishes to terminate the dialogue issues a D-END request primitive. After issuing a D-END request primitive, the DS-User must not then issue any other service primitive (except D-ABORT if required), until a D-END confirmation is received. After issuing a D-END request primitive, the DS-User must be prepared to continue receiving D-DATA indications from the peer user, until a D-END confirmation primitive is received.

Note 3.— If the D-END confirmation contains a result code of "accepted" then the dialogue no longer exists. If the D-END confirmation contains a result code of "rejected" then the peer DS-User does not wish to terminate the dialogue, and both DS-Users are then free to use the dialogue as if the D-END service had never been invoked.

Note 4.— When a DS-User receives a D-END indication primitive, it may continue to send data using the D-DATA service, but it may at some time issue a D-END response primitive, with a result code of "accepted" or "rejected". After issuing a D-END response primitive with result "accepted", a DS-User must not issue any other service primitive, as the dialogue no longer exists. After issuing a D-END response primitive with result "rejected", a DS-User may issue any other service primitive, as if the D-END service had never been invoked.

Note 5.— The parameters of the D-END primitives are specified in Table 4.2-6.

Table 4.2-6. D-END parameters

Parameter Name	*Req*	*Ind*	*Rsp*	*Cnf*
Result			*M*	*M(=)*
User Data	*U*	*C(=)*	*U*	*C(=)*

*Note 6.— The Result parameter specifies whether the requested dialogue end has been accepted. It can take one of the abstract values: "**accepted**" or "**rejected**".*

Note 7.— The User Data parameter contains the data to be transferred from a DS-User to its peer, using an existing dialogue. Its presence in the confirmation primitive is conditional upon it being specified by the DS-User in the response primitive.

Note 8.— In the event of service disruption (e.g. by D-P-ABORT), the invoker of the D-END response primitive will never know that any associated User Data failed to be delivered, as the service is already terminated.

Note 9.— A D-END collision occurs when both peers issue a D-END request primitive near-simultaneously, such that neither peer has yet received the D-END indication primitive corresponding to the remote peer's D-END request. The collision is handled by the CF on behalf of the DS-User. However, one result of the collision handling is that any User Data present in the D-END request will be delivered to the peer in a D-END confirmation primitive, rather than the usual D-END indication. This means that the peer will be unable to react to the contents of the User Data parameter, as the dialogue will have terminated. When a DS-User application is designed such that either peer may terminate the dialogue, then the application can not require a response to any User Data which is sent on a D-END request primitive. The following sequence diagram illustrates the D-END collision from the viewpoint of the two DS-Users:

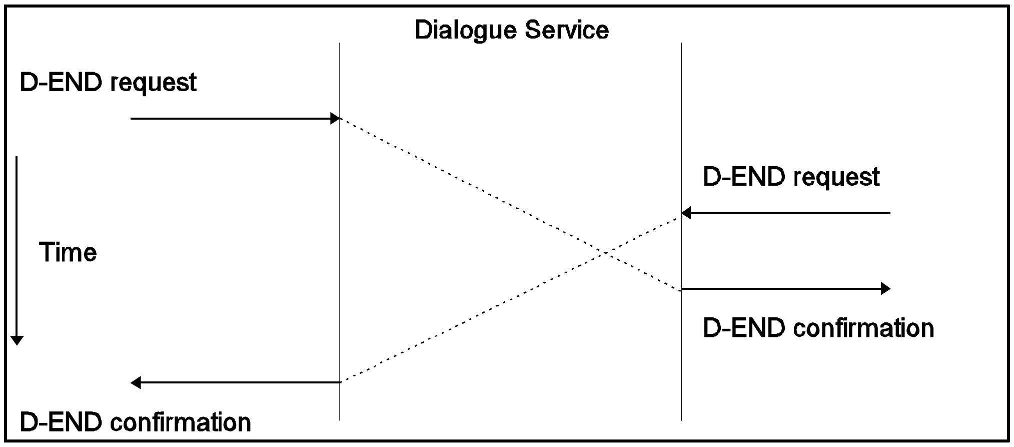


Figure 4.2-4. D-END collision sequence diagram (DS-User view)

4.2.3.5 The D-ABORT service

4.2.3.5.1 The behaviour defined by the D-ABORT service primitive shall be provided to enable the abnormal release of a dialogue between two DS-Users, by either DS-User.

Note 1.— The D-ABORT service request and indication primitives are as illustrated in Figure 4.2-5.

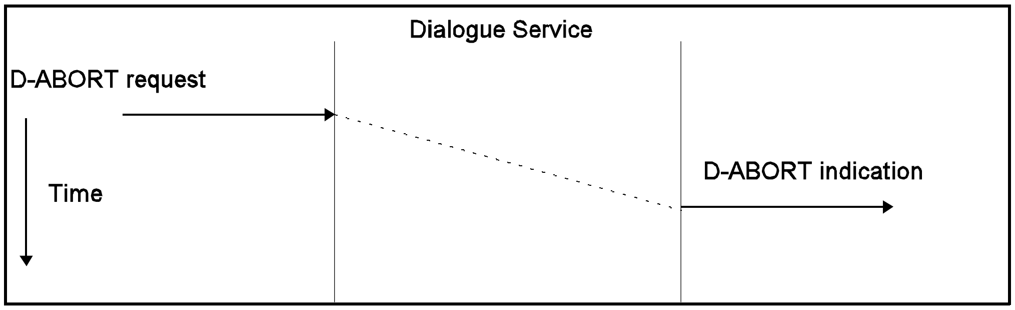


Figure 4.2-5. D-ABORT sequence diagram

Note 2.— When a dialogue is aborted, data in transfer may be lost. The parameters of the D-ABORT primitives are specified in Table 4.2-7.

Table 4.2-7. D-ABORT parameters

Parameter Name	*Req*	*Ind*
Originator	*U*	*C(=)*
User Data	*U*	*C(=)*

Note 3.— The Originator parameter is used to distinguish the source of the abort. Its presence in the indication primitive is conditional upon it being specified by the DS-User in the request primitive. It can take one of the following abstract values:

a) User — the abort originated from the Application-user; or

b) Provider — the abort originated in the ATN-App AE (including the ATN-App ASE).

Note 4.— If the D-ABORT Originator parameter is not specified, the default value "Provider" is assumed.

Note 5.— The User Data parameter contains the data to be transferred from a DS-User to its peer, using an existing dialogue. Its presence in the indication primitive is conditional upon it being specified by the DS-User in the request primitive. There is no guarantee that the peer will receive the User Data; the sender will not be informed if the User Data is not delivered.

4.2.3.6 The D-P-ABORT service

4.2.3.6.1 The behaviour defined by the D-P-ABORT service primitive shall be provided to indicate an abnormal release of a dialogue by the supporting communications service.

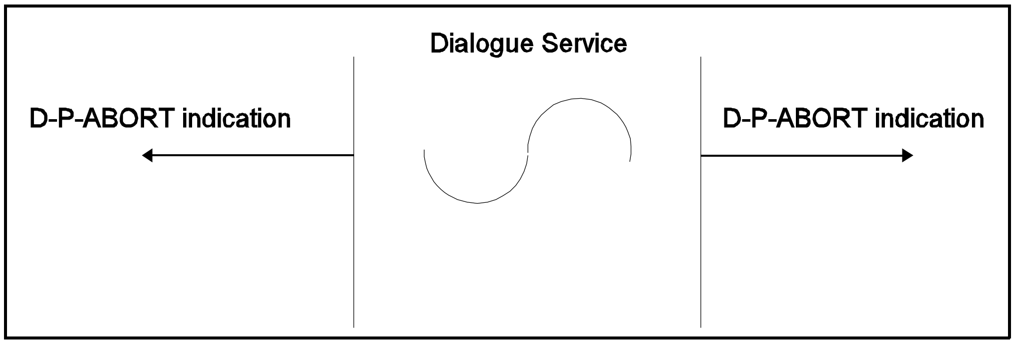


Figure 4.2-6. D-P-ABORT sequence diagram

Note 1.— For the D-P-ABORT service, only an indication primitive is defined, as illustrated in Figure 4.2-6.

Note 2.— The D-P-ABORT service allows the supporting communications service to indicate to the DS-Users that it aborted the dialogue. When the dialogue is aborted, any data in transit may be lost. The D-P-ABORT primitive has no parameters.

4.3 APPLICATION ENTITY (AE) DESCRIPTION

4.3.1 Introduction

4.3.1.1 The ATN-App AE shall exhibit external behaviour as if implemented according to the model shown in Figure 4.3-1, with the protocols defined in ACSE and the ATN-App ASE specifications.

Note 1.— As indicated in 4.1, the AE is described in terms of the Service which it displays to the Application-user, and in terms of the Control Function (CF) which mediates the interactions of the components of the AE.

Note 2.— Figure 4.3-1 also indicates which paragraph describes the behaviour of the CF in response

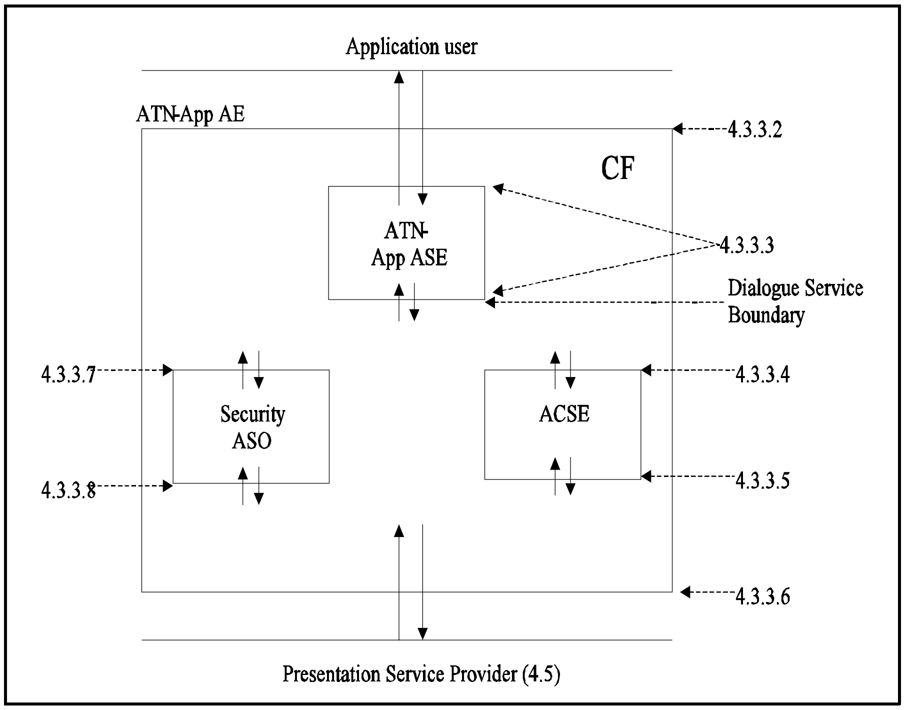


Figure 4.3-1. Components of ATN-App AE

to events at various service boundaries.

Note 3.— Additional functionality could be added in future versions of the ATN Upper Layer Architecture by the addition of further ASEs or ASOs to those shown in Figure 4.3-1. ACSE provides the basic mechanisms for establishing and releasing an application association. The service provided by the ATN-App AE represents an abstract description of the Application Programming Interface (API) seen by the Application user. The Security ASO provides security-related functions (see 4.8). The CF of the ATN-App AE specifies how the interactions at the ATN-App AE Service boundary invoke the appropriate service *primitives of the constituent ASEs, which in turn generate the actual protocol. The CF also specifies how the constituent ASEs interact with the supporting communications service.*

Note 4.— A CF specification is not a service definition of the ATN-App AE or its components. It only defines the actions of the CF as a result of service invocations visible to the CF. Thus, the specification is organised around specifying the response of the CF to these inputs. 4.3.3.2 specifies the actions that result from the inputs of the Application-user. 4.3.3.4 and 4.3.3.5 specify the actions that result from the service invocations of the ACSE component ASE. 4.3.3.3 specifies the actions that result from the service invocations of the ATN-App ASE component ASE. 4.3.3.6 specifies the actions that result from the inputs from the supporting service. 4.3.3.7 and 4.3.3.8 specify the actions that result from service invocation and APDU exchanges of the Security ASO component ASE.

Note 5.— The CF specification describes the overall behaviour of the ATN-App AE. It is not a requirement that an identifiable CF entity be realised in an implementation.

Note 6.— This CF specification assumes that the embedded ASEs (ATN-App ASE and ACSE) are modelled as atomic entities, such that when an input event is invoked by the CF, that event is processed to completion by the ASE and the CF responds to any resulting output events from the ASE, all within the same logical processing thread. This model avoids the need to specify further transient states within the CF. It does not imply any particular implementation architecture.

Note 7.— In the current version of the ATN Upper Layers, the service interface presented to the Application-user is a simple pass-through to the ATN-App ASE. That is, the Application-user passes request and response primitives directly to, and receives indication and confirmation primitives directly from, the ATN-App ASE.

Note 8.— The CF described here supports the four air-ground applications currently defined. The specification of the CF for the AIDC application is included in the AIDC specification.

Note 9.— For the purposes of this specification, the ATN-App AE is modelled such that a new instance of communication (effectively a new AE invocation) is implicitly created (a) for each request from the AE-User that will require a new association (i.e., that will result in a D-START request being invoked), and (b) for each indication from the underlying communications service that a new connection is requested. The AE invocation ceases to exist when the underlying communications service connection is disconnected and the CF is idle (i.e., in the NULL state).

Note 10.— For the purposes of this specification, when primitives or APDUs are submitted to or received from the Security ASO, the CF model requires that parameters from DS or ACSE primitive invocations are 'stored for future use' and subsequently 'retrieved.' This is to allow the CF context to be explicitly maintained over invocations of the Security ASO. This is merely a descriptive convention, and does not imply any particular implementation architecture, nor does it imply any physical storage requirements in a real system.

4.3.2 Application Level Naming and Context Definition

4.3.2.1 ATN Upper Layers Naming Hierarchy

Note 1.— Names, in the form of object identifiers (OIDs), are assigned here to the defined ATN entities.

Note 2.— The upper nodes of the ATN naming hierarchy are defined in Sub-Volume 9. The portion of the ATN naming hierarchy relevant to the Upper Layer Communications Service is illustrated in Figure 4.3-2.

Note 3.— The root nodes of the naming subtrees relevant to the Upper Layer Communications Service are shown in Table 4.3-1.

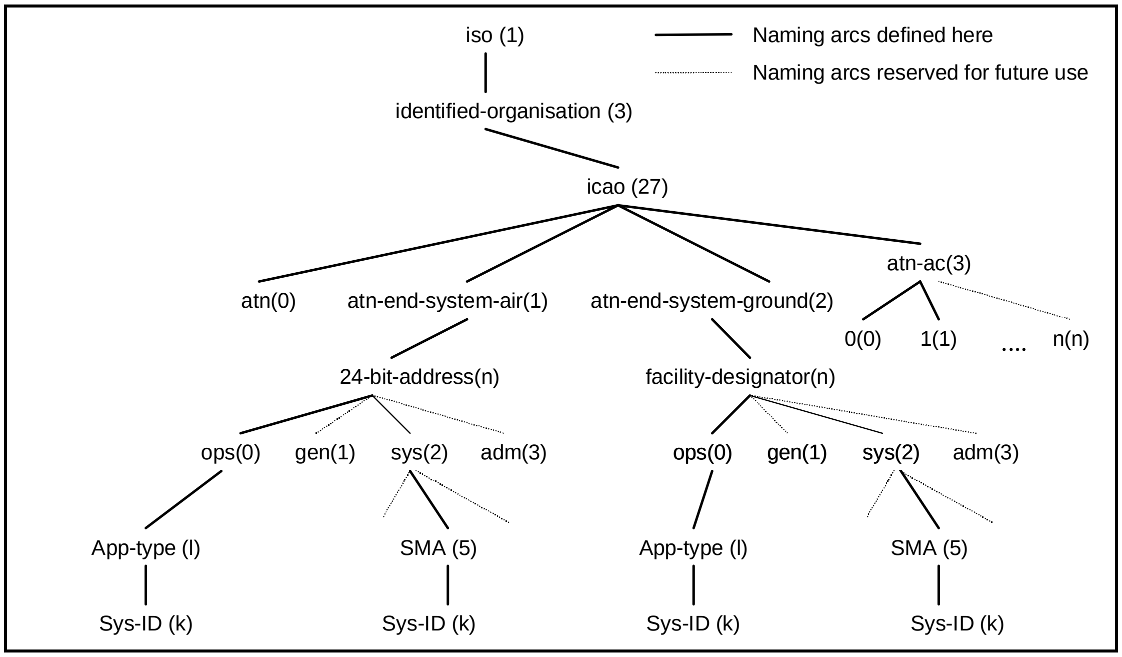


Figure 4.3-2. ATN Naming Hierarchy

Table 4.3-1. Upper Layers Naming Roots

Name and numeric value	**Description**
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

4.3.2.2 Application Process titles

Note.— Application process titles are allocated underneath either of the Object Identifier arcs:

{ atn-end-system-air (1) } or { atn-end-system-ground (2) }.

Immediately subordinate to this arc is an arc whose value is an INTEGER derived from either the 24-bit ICAO aircraft address or the ICAO facility designator, as described in 4.3.2.4. Immediately beneath that arc is an arc whose value is determined by the category of the ATN application. For the present, only the following names and values are defined for the application category:

{ ops (0) } - for operational applications, and

{ sys (2) } - for system management applications.

Subordinate to the application category arc is an arc whose value is determined by the type of the ATN application process (e.g. ads (0), cma (1), etc.), as defined in Table 9-2.

4.3.2.2.1 Each application process type on each ATN end system shall be assigned an unambiguous application process title (AP-title).

4.3.2.2.2 The AP-title shall be an Object Identifier type (i.e. an AP-title-form2 as defined in ISO/IEC 8650-1 | ITU-T Rec. X.227 (1994)).

4.3.2.2.3 Application Process titles shall be of the form:
either:
{iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) <end-system-id> (n) category (m) <app-type> (l)}
or:
{iso (1) identified-organisation (3) icao (27) atn-end-system-ground (2) <end-system-id> (n) category (m) <app-type> (l)}
where:

<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>,
	Note.— The algorithm for deriving the INTEGER n from the <end-system-id> is defined in 4.3.2.4.
<category>	is the application category, either ops (Operational) or sys (System Management),
(m)	is the INTEGER value corresponding to the application category, ops(0) or sys(2),
<app-type>	is the application type as specified in Table 9-2, and

(l) is the INTEGER value corresponding to the application type, and takes one of the values specified in Table 9-2.

4.3.2.2.4 The app-type arc of the Application Process title object identifier represents the ATN application type (e.g. “ADS” or “CMA”), and shall take one of the values specified in Table 9-2.

Note.— Table 9-2 is the global register of all standard ATN application types. Table 4.3-2 shows how the names and numeric values that are assigned in Table 9-2 are used in the construction of ATN Application Process titles.

Table 4.3-2. Examples of ATN Application Process titles

ATN ASE type	**Application Process title**
ADS-air ASE	iso(1) identified-organisation(3) icao(27) atn-end-system-air(1) <aircraft id>(n) ops(0) ads(0)
CM-ground ASE	iso(1) identified-organisation(3) icao(27) atn-end-system-ground(2) "ABCDEFGH"(n) ops(0) cma(1)
(Refer to Table 9-2 for complete list of ASE types)	...

4.3.2.3 Application Entity Titles

4.3.2.3.1 Each ATN application entity shall be assigned an unambiguous application entity title (AE Title).

4.3.2.3.2 For ATN, an AE Title shall be an Object Identifier type as defined in ISO/IEC 8824-1 | ITU-T Rec. X.680 (1995) (i.e. an AE-title-form2 as defined in ISO/IEC 8650-1 | ITU-T Rec. X.227 (1994)).

Note.— The AE Title is composed of an Application Process title (AP-title) and an AE-qualifier.

4.3.2.3.3 The AE-qualifier component of the AE Title shall be an INTEGER type (i.e. an AE-qualifier-form2 as defined in ISO/IEC 8650-1 | ITU-T Rec. X.227 (1994)).

4.3.2.3.4 The AE-qualifier value arc of the AE Title object identifier shall either be NULL (i.e. absent) or an unambiguous system identifier, with value as specified in 4.3.2.4.

Note.— The AE-qualifier further qualifies the Application Process title such that it identifies a given application type on a specific End System at a given location.

4.3.2.3.5 Thus, AE-titles conforming to this definition shall be of the form:

either:

{ iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) <end-system-id> (n) category (m) <app-type> (l) [<ae-qualifier> (k)] }

or

{ iso (1) identified-organisation (3) icao (27) atn-end-system-ground (2) <end-system-id> (n) category (m) <app-type> (l) [<ae-qualifier> (k)] }

where:

<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>,
	Note.— The algorithm for deriving the INTEGER n from the <end-system-id> is defined in 4.3.2.4.
<category>	is the application category, either ops (Operational) or sys (System Management),
(m)	is the INTEGER value corresponding to the application category, ops(0) or sys(2),
<app-type>	is the name form application type as specified in Table 9-2,
(l)	is the INTEGER value corresponding to the application type, and takes one of the values specified in Table 9-2,
<ae-qualifier>	is the unambiguous system identifier, if present, and
(k)	is the INTEGER value corresponding to the 8-octet LOC+SYS fields from the ATN NSAP address, if present.
	Note.— The algorithm for deriving the INTEGER k from the <ae-qualifier> is defined in 4.3.2.4.

4.3.2.4 Encoding of End System Identifiers and AE-qualifiers

Note 1.— Where <end-system-id> and <ae-qualifier> appear as components of an Object Identifier, the encoding of the OID subidentifier value is obtained as defined in the following text.

Note 2.— For ground stations, the <end-system-id> is derived from a four to eight character facility designator, e.g., "LFPODLHX". The syntax of the first four letters is defined in ICAO Doc 7910 "Location Indicators" (the value "0000" is reserved - see 4.3.3.8.1.2.2.e)); the syntax of the remaining letters is defined in ICAO Doc 8585 "Designators for Aircraft Operating Agencies, Aeronautical Authorities and Services."

4.3.2.4.1 For aircraft, the <end-system-id> naming arc shall be the binary value of the 24-bits comprising the ICAO aircraft identifier, expressed as an INTEGER in the range 0..(2^{24}-1) and encoded as an Object Identifier subidentifier as defined in ISO/IEC 8825-1 | ITU-T Rec. X.690 (1995).

4.3.2.4.2 For ground stations, the encoding of the <end-system-id> naming arc shall be derived from the ICAO facility designator, which is a sequence of characters from the restricted character set (A..Z), as follows:

a) Each character is encoded into one octet where:

   1) the most significant bit (bit 8) indicates whether the character is the last in the sequence: it is set to zero in the last octet and one in each preceding octet;

   2) the next most significant bit (bit 7) is set to zero;

   3) the six least significant bits (bits 6 - 1) contain the character encoded as a 6-bit value such that A is encoded as the binary value 000001, B is encoded as 000010, and so on up to Z which is encoded as 011010

   Note.— This coding gives compatibility with the Basic Encoding Rules for an Object Identifier subidentifier in ISO/IEC 8825-1 | ITU-T Rec. X.690. The character coding is equivalent to the "6-bit ASCII" subset of International Alphabet Number 5 (IA5) defined by ITU, which is adopted in SSR Mode S specifications. If required, the encoding can be extended to include numeric characters, with 0 to 9 encoded as binary values 110000 to 111001 respectively, and the space character can be encoded as binary value 100000.

b) The <end-system-id> is the concatenation of these octets.

Note.— Conceptually, bits 7 -1 from each octet are concatenated to form an unsigned binary number whose most significant bit is bit 7 of the first octet and whose least significant bit is bit 1 of the last octet.

4.3.2.4.3 When expressed as the final arc of the AE Title OID, the value of <ae-qualifier> shall consist of the binary value of the 8-octets comprising the LOC+SYS fields of the ATN NSAP address, with the LOC octets as the most significant, expressed as an INTEGER in the range 0..(2^{64}-1) and encoded as an Object Identifier subidentifier as defined in ISO/IEC 8825-1 | ITU-T Rec. X.690 (1995).

4.3.2.4.4 When used as an INTEGER value in ACSE primitives and APDUs, the <ae-qualifier> shall be taken as a signed value, in the range -2^{63} to $+(2^{63} - 1)$, that is the concatenation of the bits of the 2-octet LOC and the 6-octet SYS fields, with the LOC bits as the most significant, encoded as an unconstrained INTEGER value.

Note.— This means that, if the leading bit of LOC has the value 1, the PER-encoded value for Sys-ID will equate to a negative integer value.

4.3.2.5 Application Context Names

Note 1.— The Application Context describes the ASE/ASO types which are present in the AE, including those aspects not distinguished by ASO type (e.g. version and policy aspects). The abstract syntax of the APDUs and the control function are described here. The Application Context name is an identifier which is used to refer to a defined Application Context. The syntax of the Application Context name is defined in ISO/IEC 8650-1 | ITU-T Rec. X.227 as an Object Identifier.

Note 2.— The application context name is used here only to distinguish between different versions of an application context within the scope of a given AE type, as identified by the AE Title.

4.3.2.5.1 The Application Context name shall be used to indicate the version and policy aspects relative to the AE with which it is associated.

4.3.2.5.2 Each Application Context shall be assigned an Application Context name.

4.3.2.5.3 Application Context names shall have the following structure:

{iso (1) identified-organisation (3) icao (27) atn-ac (3) version-<n> (n)}
where n is an INTEGER in the range 0..255.

Note.— The value n = 0 is reserved for use by the CF.

4.3.2.6 Presentation Context Identification

Note.— The Null Encoding presentation protocol option has been selected for the most efficient encoding of presentation PDUs, as defined in 4.5. As a consequence, the conventional presentation protocol mechanisms which enable users of the presentation service to distinguish the presentation context of received APDUs are not available. Therefore, an alternative, application layer, mechanism is defined here.

4.3.2.6.1 All User Data which is passed across the presentation service boundary shall be encoded using the unaligned variant of the Packed Encoding Rules (PER) for ASN.1 (ISO/IEC 8825-2 | ITU-T Rec. X.691 (1995)).

4.3.2.6.2 When in the data transfer phase, in order to be able to distinguish APDUs which are defined in different abstract syntax modules, the presentation User Data encoding shall assume the Full Encoding option of ISO/IEC 8823-1 | ITU-T Rec. X.226 (1994), augmented with the PER-visible constraints defined in ISO/IEC 8823-1: 1994/Amd. 1: 1997 | ITU-T Rec. X.226 (1994)/Amd.1 (1997) as follows:

```
Fully-encoded-data ::= SEQUENCE SIZE (1, ...) OF PDV-list
-- contains one or more presentation-data-value-list (PDV-list) values
PDV-list ::= SEQUENCE
{ transfer-syntax-name        Transfer-syntax-name OPTIONAL,
 presentation-context-identifier    Presentation-context-identifier,
 presentation-data-values  CHOICE
           { single-ASN1-type [0] ABSTRACT-SYNTAX.&Type
                     (CONSTRAINED BY {
                     -- Type corresponding to presentation context identifier -- }) ,
           octet-aligned            [1] IMPLICIT OCTET STRING,
           arbitrary           [2] IMPLICIT BIT STRING }
-- contains one or more presentation data values from the same
-- presentation context.
           }

Transfer-syntax-name    ::= OBJECT IDENTIFIER -- not used for ATN Upper Layers

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

Note 1.— Note that ISO/IEC 8823-1 | ITU-T Rec. X.226 specifies two choices for the encoding of User-data, either Simply-encoded-data or Fully-encoded-data. For ATN, presentation User Data is equivalent to Fully-encoded-data, and NOT to ISO/IEC 8823-1 | ITU-T Rec. X.226 User-data. That is, the bit to indicate the CHOICE of simple or full encoding is NOT encoded.

Note 2.— The use of Full Encoding is specified in order to overcome the fact that: (a) the use of presentation protocol efficiency enhancements removes the ability of presentation layer to perform the necessary demarcation, and (b) the use of ASN.1 Packed Encoding Rules means that it would not have been possible to assign unique ASN.1 tag values to individual APDUs to distinguish between them, as PER does not encode tags.

4.3.2.6.3 Only the presentation-context-identifier and presentation-data-values fields shall be present in the encoded presentation User Data.

4.3.2.6.4 Only the “arbitrary” (BIT STRING) choice for presentation-data-values in the PDV-list SEQUENCE shall be used in the encoded presentation User Data.

4.3.2.6.4 bis A bit-oriented encoding shall be applied, such that no padding bits are appended to the encoded BIT STRING value, and the length determinant of the BIT STRING encoding equates to the number of significant bits.

Note.— The above provision means that data encoded by ATN ASEs, when embedded in presentation-data-values, are treated by the CF as normal BIT STRING values, not in general an integral number of octets. Padding to an octet boundary only applies to the outermost Fully-encoded-data value that is passed across the Presentation Service boundary.

4.3.2.6.5 The values of Presentation-context-identifier which are pre-defined in Table 4.3-3 shall be used in the encoding of presentation User Data; the presentation-context-identifiers are not dynamically assigned by the presentation service.

Table 4.3-3. Presentation Context identifiers

Presentation-context-identifier value	**Short name**	**Description**
1	acse-apdu	ACSE abstract syntax as defined in ISO/IEC 8650-1 /Amd.1 | ITU-T Rec. X.227 (1994)/Amd.1 (1996)
2		reserved for future use
3	user-ase-apdu	abstract syntax as defined in individual ATN application specifications
other		reserved for future use

4.3.2.6.6 With the sole exception of the P-CONNECT service (which is used exclusively by ACSE), upon receiving User Data from the presentation service, the CF shall:

a) decode the Fully-encoded-data and use the presentation-context-identifier value to determine the target ASE.

b) if the target ASE is ACSE, decode the header of the embedded presentation-data-value to determine the APDU type, and

c) if the decoding in a) and b) fails for any reason (presentation-context-identifier not recognised, presentation-data-value does not use the “arbitrary” CHOICE value, or unrecognised APDU type) then issue a P-U-ABORT request to the supporting service and behave as if a P-U-ABORT indication with no parameters has been received;

d) otherwise, pass the presentation-data-value (i.e., acse-apdu or user-ase-apdu) to the target ASE by invoking the appropriate indication or confirmation primitive at the lower ASE service boundary, as specified in 4.3.3.

4.3.2.6.7 Except for P-CONNECT primitives issued by ACSE, when an ASE issues a request or response primitive at its lower service boundary which would otherwise map onto a presentation service primitive, the CF shall:

a) embed the User Data into a Fully-encoded-data type, using the presentation-context-identifier value corresponding to the source ASE

b) pass the Fully-encoded User Data to the presentation service by invoking the appropriate primitive, as specified in 4.3.3.

4.3.2.6.8 Between peer ATN-App AEs, a single default presentation context shall be used; this is known by bilateral agreement.

Note 1.— All component abstract syntax modules (e.g. ATN-App-ASE, ACSE, SESE) are considered merged into one. It is the job of the CF to merge and split contexts, since ATN does not use presentation layer context handling services.

Note 2.— ASN.1 packed encoding rules, basic unaligned variant, are used to provide the transfer syntax for the single abstract syntax.

Note 3.— Clause 7.9 of the PER Standard is not applicable to ATN, since:

a) EXTERNAL values are fully resolved since all abstract syntaxes are known a priori,

b) when carried in the (null) presentation protocol, 'full encoding' with the BIT STRING choice alternative is used.

Note 4.— The 'outermost value' referred to in Clause 10.1.1 of the PER Standard is interpreted in ATN context as the encoded data that passes over the Presentation Service boundary. Padding bits are appended to achieve octet alignment at this boundary.

4.3.3 Control Function Specification

4.3.3.1 ATN-App CF State Definitions

4.3.3.1.1 The ATN-App AE shall behave as if it has a Control Function which can exist only in one of the following states:

a) Null (STA0) — This is the state of the CF when there is no association in existence.

b) Association Pending (STA1) — The CF enters this state either when the ATN-App ASE has made a request to establish a dialogue and is waiting for notification from its peer OR an indication has been received that the peer has made a request to establish a dialogue.

c) Data Transfer (STA2) — The CF enters this state once the establishment phase is complete. An association has successfully been established and the communicating partners are free to send and receive data.

d) Release Pending (STA3) — The CF enters this state when the ATN-App ASE has requested the termination of the dialogue OR an indication has been received that the peer has made a request to terminate the dialogue.

e) Release Collision (STA4) — The CF enters this state when both communicating partners have requested the termination of the dialogue near-simultaneously.

4.3.3.1.2 CF State Table

4.3.3.1.2.1 The ATN-App AE CF shall behave as if it has a control function in accordance with the state table specified in Table 4.3-4, which shows diagrammatically the state transitions and actions performed by the CF in response to incoming events.

Note.— The following conventions are used in Table 4.3-4:

a) Incoming events are shown in the first two columns of the state table, and are enumerated in Table 4.3-6.

b) When an input event occurs and the state table indicates an action, the CF performs that action.

c) Each cell in the state table shows:

1) optionally, one or more predicates, denoted "pN", where N is an integer. The state and action which follow the predicate are only valid if the predicate is TRUE. The inverse (logical NOT) of a predicate is indicated by the prefix "~" (tilde *character), the combination (logical AND) of two or more predicates is indicated by the symbol "&" (ampersand), and the choice of two or more predicates (logical OR) is indicated by the symbol "|" (vertical bar).*

2) *the new state that the CF enters after the action has been performed*

3) *the action, if any, which the CF performs. The possible actions are outlined in Table 4.3-7.*

d) *Blank cells indicate error conditions.*

e) *When an input event occurs and the state table indicates a state transition, the CF enters the new state after any associated action has been performed.*

4.3.3.1.2.2 For the purpose of specifying CF behaviour, embedded ASEs (ATN-App ASE and ACSE) shall be treated as atomic entities, such that when an input event is invoked by the CF, that event is processed to completion by the ASE and the CF responds to any resulting output events from the ASE, all within the same logical processing thread.

Note.— This provision avoids the need to specify further transient states within the CF. It does not imply any particular implementation architecture.

4.3.3.1.2.3 The following combinations of input events and CF states shall be treated as error conditions:

a) The occurrence of an input event other than those listed in Table 4.3-6; or

b) A combination of input event and CF state which corresponds to a blank cell in Table 4.3-4; or

c) A combination of input event and CF state which corresponds to a cell in Table 4.3-4 containing one or more predicates, none of which evaluates to TRUE.

4.3.3.1.2.4 The error handling shall result in the association being aborted, if one exists, and a notification being given to the Application user.

4.3.3.1.2.5 In the event of a conflict between the actions implied by the state table and the text in the following paragraphs, the text shall take precedence.

Table 4.3-4. ATN-App CF State Table

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
From ATN-App	ATN-APP function req	STA0 ATN-App ASE req	STA1 ATN-App ASE req	STA2 ATN-App ASE req	STA3 ATN-App ASE req	STA4 ATN-App ASE req
	ATN-APP function rsp	STA0 ATN-App ASE rsp	STA1 ATN-App ASE rsp	STA2 ATN-App ASE rsp	STA3 ATN-App ASE rsp	STA4 ATN-App ASE rsp
From ATN-App ASE (upper)	ATN-APP function ind	STA0 ATN-App ind	STA1 ATN-App ind	STA2 ATN-App ind	STA3 ATN-App ind	STA4 ATN-App ind
	ATN-APP function cnf	STA0 ATN-App cnf	STA1 ATN-App cnf	STA2 ATN-App cnf	STA3 ATN-App cnf	STA4 ATN-App cnf
From ATN-App ASE (lower)	D-START req	p0 & p4: STA1 SA-START req p0 & p5: STA1 SA-SEND req p0 & ~p6: STA1 A-ASSOC req				
	D-START rsp+		~p1 & p6 & p4: STA1 SA-START rsp ~p1 & p6 & p5: STA1 SA-SEND req ~p1 & ~p6: STA1 A-ASSOC rsp+			
	D-START rsp-		~p1 & p6 & p4: STA1 SA-START rsp ~p1 & p6 & p5: STA1 SA-SEND req ~p1 & ~p6: STA1 A-ASSOC rsp-			

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
	D-DATA req			p4 | p5: STA2 SA-SEND req ~(p4 | p5): STA2 P-DATA req (User)	~p2 & (p4 | p5): STA3 SA-SEND req ~p2 & ~(p4 | p5): STA3 P-DATA req (User)	
	D-END req			p4 | p5: STA3 SA-SEND req ~(p4 | p5): STA3 A-RELEASE req		
	D-END rsp+				~p2 & (p4 | p5): STA3 SA-SEND req ~p2 & ~(p4 | p5) STA3 A-RELEASE rsp+	
	D-END rsp-				~p2 & (p4 | p5): STA3 SA-SEND req ~p2 & ~(p4 | p5): STA3 A-RELEASE rsp-	
	D-ABORT req		STA1 A-ABORT req	p4 | p5: STA2 SA-SEND req ~(p4 | p5): STA2 A-ABORT req	STA3 A-ABORT req	STA4 A-ABORT req

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
From ACSE (upper)	A-ASSOCIATE ind		p6: STA1 SASO-deliver ~p6: STA1 D-START ind			
	A-ASSOCIATE cnf+		p6 & (p4 | p5): STA 1 SASO-deliver p6 & ~(p4 | p5): STA2 D-START cnf+ ~p6: STA2 D-START cnf+			
	A-ASSOCIATE cnf-		p6 & ~(p4 | p5): STA0 D-START cnf- p6 & (p4 | p5): STA1 SASO-deliver ~p6: STA0 D-START cnf-			
	A-RELEASE ind				p4 | p5: STA3 SASO-deliver ~(p4 | p5): STA3 D-END ind	p4 | p5: STA4 SASO-deliver ~(p4 | p5) & p1: STA4 A-RELEASE rsp+ ~(p4 | p5) & ~p1: STA4
	A-RELEASE cnf+				p4 | p5: STA3 SASO-deliver ~(p4 | p5): STA0 D-END cnf+ P-U-ABORT req	p4 | p5: STA4 SASO-deliver ~(p4 | p5) & p1: STA0 D-END cnf+ P-U-ABORT req ~(p4 | p5) & ~p1: STA4 D-END cnf+ A-RELEASE rsp+

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
	A-RELEASE cnf-				p4 | p5: STA3 SASO-deliver ~(p4 | p5): STA2 D-END cnf-	
	A-ABORT ind		STA0 D-ABORT ind	p4 | p5: STA2 SASO-deliver ~(p4 | p5): STA0 D-ABORT ind	STA0 D-ABORT ind	STA0 D-ABORT ind
	A-P-ABORT ind		STA0 D-P-ABORT ind	STA0 D-PABORT ind	STA0 D-P-ABORT ind	STA0 D-P-ABORT ind
From ACSE (lower)	P-CONNECT req		STA1 P-CONN req			
	P-CONNECT rsp+		STA2 P-CONN rsp+			
	P-CONNECT rsp-		STA0 P-CONN rsp-			
	P-RELEASE req				STA3 P-DATA req (RLRQ)	
	P-RELEASE rsp+				STA0 P-DATA req (RLRE+)	p1: STA4 P-DATA req (RLRE+) ~p1: STA0 P-DATA req (RLRE+)
	P-RELEASE rsp-				STA2 P-DATA req (RLRE-)	
	P-U-ABORT req (data)	STA0 P-U-ABORT req	STA0 P-U-ABORT req	STA0 P-DATA req (ABRT)	STA0 P-U-ABORT req	STA0 P-U-ABORT req
	P-U-ABORT req (no data)		STA0 P-UABORT req	STA0 P-U-ABORT req	STA0 P-U-ABORT req	STA0 P-U-ABORT req

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
From supporting service	P-CONNECT ind	p0: STA1 P-CONN ind				
	P-CONNECT cnf+		STA1 P-CONN cnf+			
	P-CONNECT cnf-		STA1 P-CONN cnf-			
	P-DATA ind (RLRQ)	p3: STA0		STA3 P-RELEASE ind	p2: STA4 P-RELEASE ind	
	P-DATA ind (RLRE+)	p3: STA0			STA3 P-RELEASE cnf+	STA4 P-RELEASE cnf+
	P-DATA ind (RLRE-)	p3: STA0			STA3 P-RELEASE cnf-	
	P-DATA ind (ABRT)	p3: STA0 P-U-ABORT req ~p3: STA0		STA2 P-U-ABORT ind P-U-ABORT req	STA3 P-U-ABORT ind P-U-ABORT req	STA4 P-U-ABORT ind P-U-ABORT req
	P-DATA ind (User)	p3: STA0		p4 | p5: STA2 SASO-deliver ~(p4 | p5): STA2 D-DATA ind	p2 & (p4 | p5): STA3 SASO-deliver p2 & ~(p4 | p5): STA3 D-DATA ind	
	P-U-ABORT ind	STA0	STA1 P-U-ABORT ind	STA2 P-U-ABORT ind	STA3 P-U-ABORT ind	STA4 P-U-ABORT ind
	P-P-ABORT ind	STA0	STA1 P-P-ABORT ind	STA2 P-P-ABORT ind	STA3 P-P-ABORT ind	STA4 P-P-ABORT ind

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
From Security ASO (Upper)	SA-START ind		p4: STA1 D-START ind			
	SA-START cnf		p11 & p4: STA0 D-START cnf- p9 & p4: STA2 D-START cnf+			

Event Source	**State-->** **Event**	**STA0** **Null**	**STA1** **Assoc. Pending**	**STA2** **Data Transfer**	**STA3** **Release Pending**	**STA4** **Release Collision**
	SA-SEND ind		p1 & p5 & p9: STA2 D-START cnf+ p1 & p5 & p11: STA0 D-START cnf- ~p1 & p5: STA1 D-START ind	~p7: STA2 D-DATA ind p7: STA0 D-ABORT ind	~p2: STA3 D-END ind p2 & ~(p9 | p11): STA3 D-DATA ind p2 & p11: STA2 D-END cnf- p2 & p9: STA0 D-END cnf+ P-UABORT req	p1 & ~(p9 | p11): STA4 SA-SEND req ~p1 & ~(p9 | p11): STA4 p1 & p9: STA0 D-END cnf+ P-UABORT req ~p1 & p9: STA4 D-END cnf+ SA-SEND req
From Security ASO (Lower)	SASO-APDU	STA0	p1: STA1 A-ASSOC req ~p1 & p9: STA1 A-ASSOC rsp+ ~p1 & p11: STA1 A-ASSOC rsp-	~p7: STA2 P-DATA req p7: STA2 A-ABORT req	p2: STA3 A-RELEASE req ~p2 & p9: STA3 A-RELEASE rsp+ ~p2 & p11: STA3 A-RELEASE rsp- ~p2 & ~(p9 |p11): STA3 P-DATA req	STA4 A-RELEASE rsp+

Table 4.3-5. Predicates used in Table 4.3-4

Predicate	**Meaning**
p0	This is a new instance of communication, i.e., no previous association exists (effectively, a new AE invocation is created).
p1	This CF is the initiator CF, i.e., the CF which issued an A-ASSOCIATE request primitive.
~p1	This CF is the responder CF, i.e., the CF which received an A-ASSOCIATE indication primitive.
p2	This CF is the Release Initiator, i.e., the CF issued an A-RELEASE request primitive.
~p2	This CF is the Release Responder, i.e., the CF received an A-RELEASE indication primitive.
p3	This CF is the “Abort+Data” initiator, i.e., the CF issued a P-DATA request containing an ABRT APDU and is awaiting disconnection by the peer.

Predicate	**Meaning**
~p3	This CF has not initiated an Abort containing user data.
p4	Dialogue supporting key management exchange
p5	Secured dialogue
p6	Security initially required on the Dialogue
p7	Dialogue aborting (D-ABORT req or A-ABORT ind has been issued
p9	Positive confirmation/response received.
p11	Negative confirmation/response received.

Table 4.3-6. Incoming Event List

Abbreviated name	**Source**	**Description**
ATN-APP function req	upper AE service boundary	Application-specific Request primitive issued by the Application User
ATN-APP function rsp		Application-specific Response primitive issued by the Application User
ATN-APP function ind	ATN-App ASE (upper service boundary)	Application-specific Indication primitive issued by the Application ASE
ATN-APP function cnf		Application-specific Confirmation primitive issued by the Application ASE
D-START req	ATN-App ASE (lower service boundary)	D-START Request primitive issued by DS-User
D-START rsp+		D-START Response primitive issued by DS-User, with Result = accepted
D-START rsp-		D-START Response primitive issued by DS-User, with Result = rejected (transient) or rejected (permanent)
D-DATA req		D-DATA Request primitive issued by DS-User
D-END req		D-END Request primitive issued by DS-User
D-END rsp+		D-END Response primitive issued by DS-User, with Result = accepted
D-END rsp-		D-END Response primitive issued by DS-User, with Result = rejected
D-ABORT req		D-ABORT Request primitive issued by DS-User
A-ASSOCIATE ind	ACSE (upper service boundary)	A-ASSOCIATE Indication primitive issued by ACSE service
A-ASSOCIATE cnf+		A-ASSOCIATE Confirmation primitive issued by ACSE service, with Result = accepted
A-ASSOCIATE cnf-		A-ASSOCIATE Confirmation primitive issued by ACSE service, with Result = rejected (transient) or rejected (permanent)

Abbreviated name	**Source**	**Description**
A-RELEASE ind		A-RELEASE Indication primitive issued by ACSE service
A-RELEASE cnf+		A-RELEASE Confirmation primitive issued by ACSE service, with Result = affirmative
A-RELEASE cnf-		A-RELEASE Confirmation primitive issued by ACSE service, with Result = negative
A-ABORT ind		A-ABORT Indication primitive issued by ACSE service
A-P-ABORT ind		A-P-ABORT Indication primitive issued by ACSE service
P-CONNECT req	ACSE (lower service boundary)	P-CONNECT Request primitive issued by ACSE Protocol Machine (ACPM)
P-CONNECT rsp+		P-CONNECT Response primitive issued by ACPM, with Result = acceptance
P-CONNECT rsp-		P-CONNECT Response primitive issued by ACPM, with Result = user-rejection or provider-rejection
P-RELEASE req		P-RELEASE Request primitive issued by ACPM
P-RELEASE rsp+		P-RELEASE Response primitive issued by ACPM, with Result = affirmative
P-RELEASE rsp-		P-RELEASE Response primitive issued by ACPM, with Result = negative
P-U-ABORT req (data)		P-U-ABORT Request primitive issued by ACPM, with the User Data parameter present.
P-U-ABORT req (no data)		P-U-ABORT Request primitive issued by ACPM, with the User Data parameter empty or absent.
P-CONNECT ind	supporting service	P-CONNECT Indication primitive issued by presentation service provider
P-CONNECT cnf+		P-CONNECT Confirmation primitive issued by presentation service provider, with Result = acceptance
P-CONNECT cnf-		P-CONNECT Confirmation primitive issued by presentation service provider, with Result = user-rejection or provider-rejection
P-DATA ind (RLRQ)		P-DATA Indication primitive issued by presentation service provider, with a RLRQ APDU as User-Data
P-DATA ind (RLRE+)		P-DATA Indication primitive issued by presentation service provider, with a RLRE APDU as User-Data, with the reason field set to “normal”
P-DATA ind (RLRE-)		P-DATA Indication primitive issued by presentation service provider, with a RLRE APDU as User-Data, with the reason field set to “not-finished”
P-DATA ind (ABRT)		P-DATA Indication primitive issued by presentation service provider, with an ABRT APDU as User-Data

Abbreviated name	Source	Description
P-DATA ind (User)		P-DATA Indication primitive issued by presentation service provider, with an ATN-APP APDU (e.g. an ADS-ASE protocol data unit) as User-Data
P-U-ABORT ind		P-U-ABORT Indication primitive issued by presentation service provider
P-P-ABORT ind		P-P-ABORT Indication primitive issued by presentation service provider
SA-START ind	Security ASO (upper service boundary)	SA-START Indication primitive issued by security ASO
SA-START cnf+		SA-START confirmation positive primitive issued by security ASO
SA-START cnf-		SA-START confirmation negative primitive issued by security ASO
SA-SEND ind		SA-SEND indication primitive issued by security ASO
SASO-APDU	Security ASO (lower service boundary)	APDU emitted by the Security ASO

Table 4.3-7. Outgoing Event List

Abbreviated name	Target	Description
ATN-App ind	upper AE service boundary	Application-specific Indication primitive mapped transparently from the upper service boundary of the ATN-App ASE.
ATN-App cnf		Application-specific Confirmation primitive mapped transparently from the upper service boundary of the ATN-App ASE.
ATN-App ASE req	upper ATN-App ASE service boundary	Application-specific Request primitive mapped transparently from the upper AE service boundary
ATN-App ASE rsp		Application-specific Response primitive mapped transparently from the upper AE service boundary.
D-START ind	DS-User	D-START Indication primitive issued.
D-START cnf+		D-START Confirmation primitive issued, with the Result parameter set to the abstract value “accepted”
D-START cnf-		D-START Confirmation primitive issued, with the Result parameter set to the abstract value “rejected (transient)” or “rejected (permanent)”, according to the A-ASSOCIATE Confirmation primitive which was received.
D-DATA ind		D-DATA Indication primitive issued.
D-END ind		D-END Indication primitive issued.
D-END cnf+		D-END Confirmation primitive issued, with the Result parameter set to the abstract value “accepted”.
D-END cnf-		D-END Confirmation primitive issued, with the Result parameter set to the abstract value “rejected”.
D-ABORT ind		D-ABORT Indication primitive issued.
D-P-ABORT ind		D-P-ABORT Indication primitive issued.
A-ASSOC req	ACSE service provider	A-ASSOCIATE Request primitive issued
A-ASSOC rsp+		A-ASSOCIATE Response primitive issued, with Result = “accepted”

Abbreviated name	Target	Description
A-ASSOC rsp-		A-ASSOCIATE Response primitive issued, with Result = “rejected (transient)” or “rejected (permanent)”, according to the D-START response primitive which was received.
A-RELEASE req		A-RELEASE Request primitive issued.
A-RELEASE rsp+		A-RELEASE Response primitive issued, with Result = “affirmative” and Reason = “normal”
A-RELEASE rsp-		A-RELEASE Response primitive issued, with Result = “negative” and Reason = “not-finished”
A-ABORT req		A-ABORT Request primitive issued.
P-CONN ind	lower ACSE service boundary	P-CONNECT Indication primitive invoked.
P-CONN cnf+		P-CONNECT Confirmation primitive invoked, with the Result parameter set to “acceptance”.
P-CONN cnf-		P-CONNECT Confirmation primitive invoked, with the Result parameter set to “user-rejection”.
P-RELEASE ind		P-RELEASE Indication primitive invoked.
P-RELEASE cnf+		P-RELEASE Confirmation primitive invoked, with the Result parameter set to “affirmative”.
P-RELEASE cnf-		P-RELEASE Confirmation primitive invoked, with the Result parameter set to “negative”.
P-U-ABORT ind		P-U-ABORT Indication primitive invoked.
P-P-ABORT ind		P-P-ABORT Indication primitive invoked.
P-CONN req	supporting service	P-CONNECT Request primitive issued.
P-CONN rsp+		P-CONNECT Response primitive issued, with the Result parameter set to “acceptance”.
P-CONN rsp-		P-CONNECT Response primitive issued, with the Result parameter set to “user-rejection”.
P-DATA req (RLRQ)		P-DATA Request primitive issued. The User Data parameter contains a RLRQ APDU.
P-DATA req (RLRE+)		P-DATA Request primitive issued. The User Data parameter contains a RLRE APDU, with the reason field set to “normal”.
P-DATA req (RLRE-)		P-DATA Request primitive issued. The User Data parameter contains a RLRE APDU, with the reason field set to “not-finished”.
P-DATA req (ABRT)		P-DATA Request primitive issued. The User Data parameter contains an ABRT APDU, with a non-empty user-information field.
P-DATA req (User)		P-DATA Request primitive issued. The User Data parameter contains an ATN-App ASE APDU (e.g. an ADS-ASE protocol data unit)
P-U-ABORT req		P-U-ABORT Request primitive issued.
SA-START req	upper Security ASO service boundary	SA-START request primitive issued
SA-START rsp+		SA-START positive response primitive issued
SA-START rsp-		SA-START negative response primitive issued
SA-SEND req		SA-SEND request primitive issued

Abbreviated name	**Target**	**Description**
SASO-deliver	lower Security ASO service boundary	APDU delivered to the Security ASO. The APDU was created by the peer Security ASO and conveyed via A-ASSOCIATE, A-RELEASE, A-ABORT or P-DATA.

4.3.3.2 Services Invoked by the Application User

Note 1.— The actions that result from inputs generated by the user of this ATN-App AE (see Figure 4.3-1) are defined here.

Note 2.— The service primitives available to the Application User are specific to the ATN application. This service is detailed in the individual application specifications.

4.3.3.2.1 When Invoked

4.3.3.2.1.1 Invocations of Application User Request and Response primitives by the Application-user shall be allowed when the CF is in any valid state.

4.3.3.2.2 Action Upon Invocation

4.3.3.2.2.1 When the Application User Request or Response primitive is issued, the CF shall:

a) Invoke the equivalent primitive of the ATN-App ASE service, with a one-to-one mapping of parameters; and

b) Remain in its current state.

4.3.3.3 Services Invoked by ATN-App ASE

4.3.3.3.1 ATN-App ASE Indication and Confirmation primitives

4.3.3.3.1.1 When Invoked

4.3.3.3.1.1.1 Invocations of ATN-App ASE Indication and Confirmation primitives by the ATN-App ASE shall be allowed when the CF is in any valid state.

4.3.3.3.1.2 Action Upon Invocation

4.3.3.3.1.2.1 When the ATN-App ASE Indication or Confirmation primitive is issued, the CF shall:

a) Invoke the equivalent primitive of the Application-user service with a one-to-one mapping of parameters; and

b) Remain in its current state.

4.3.3.3.2 D-START Request primitive

4.3.3.3.2.1 When Invoked

4.3.3.3.2.1.1 When the D-START Request primitive is invoked by the ATN-App ASE, a new instance of communication shall be created, with its CF initially in the NULL state.

4.3.3.3.2.2 Action Upon Invocation

4.3.3.3.2.2.1 When the D-START Request is validly invoked with the Security Requirements parameter absent, or set to the abstract value “No Security”, the CF shall:

a) Determine the app-type as defined for the ATN-App AE,

b) Construct the Application Context name, with the value of the "version" arc set equal to the DS-User Version Number parameter if provided, and set to zero otherwise,

c) If not specified in the request primitive, retrieve the local Calling Presentation Address,

d) Determine the Called Presentation Address either directly from the Called Presentation Address parameter if present, or via look-up from the Called Peer ID and Called Sys-ID parameters,

e) If the Calling Peer Id parameter is present, then retrieve the corresponding Calling AP Title. If, in addition to Calling Peer ID, the optional Calling Sys-ID parameter is present, then retrieve the corresponding Calling AE-Qualifier. If Calling Peer ID is not present, then Calling AP Title and Calling AE-Qualifier are not used in the A-ASSOCIATE request (and they will not then be included in the resulting A-ASSOCIATE-REQUEST (AARQ) APDU),

Note.— The way that the Calling AP Title and the Calling AE-Qualifier are retrieved is a local implementation matter.

f) Make no use of the A-ASSOCIATE parameter “ACSE Requirements”.

g) Construct an A-ASSOCIATE Request primitive with the following parameters:

Table 4.3-8.

A-ASSOCIATE Request parameter	ISO Status	ATN value
Mode	U	Not used (default value)
Application Context Name	M	As derived in b) above
Application Context Name List	C	Not used
Calling AP Title	U	As derived in e) above
Calling AE Qualifier	U	As derived in e) above
Calling AP Invocation-identifier	U	Not used
Calling AE Invocation-identifier	U	Not used
Called AP Title	U	Not used
Called AE Qualifier	U	Not used
Called AP Invocation-identifier	U	Not used
Called AE Invocation-identifier	U	Not used
ACSE Requirements	U	As derived in f) above
Authentication-mechanism-name	U	Not used
Authentication-value	U	Not used
User Information	U	D-START User Data parameter
Calling Presentation Address	M	Derived as in c) above
Called Presentation Address	M	Derived as in d) above
Presentation Context Definition List	U	Not used
Default Presentation Context Name	U	Not used
Quality of Service	M	See 4.3.3.3.2.3
Presentation Requirements	U	Not used (default value)
Session Requirements	M	No Orderly Release (NOR), Duplex
Initial Synchronization Point Serial No	C	Not used
Initial Assignment of Tokens	C	Not used
Session-connection Identifier	U	Not used

h) Invoke the A-ASSOCIATE Request primitive

i) Enter the ASSOCIATION PENDING state as an initiator CF.

4.3.3.3.2.2.2 When the D-START Request is validly invoked with the Security Requirements parameter set to the abstract value “Secured Dialogue Supporting Key Management”, the CF shall:

a) Retrieve the called and calling Entity IDs,

Note 1.— The syntax of the SA-START parameters is described in 4.8. The way that the Calling AP Title and the Calling AE-Qualifier are retrieved is a local implementation matter.

b) Construct a SA-START Request primitive with the following parameters:

Table 4.3-9.

SA-START Request parameter	**ATN value**
Called Entity ID	as specified in a) above
Calling Entity ID	as specified in a) above
User Data	D-START User Data parameter

c) Store the D-START Request parameters for future use

Note 2.— The parameters will be required after the Security ASO finishes processing the SA-START Request by issuing an APDU to send via A-ASSOCIATE.

d) Submit the SA-START Request primitive to the Security ASO (see 4.8.3.2)

e) Enter the ASSOCIATION PENDING state as an initiator CF.

4.3.3.3.2.2.3 When the D-START Request is validly invoked with the Security Requirements parameter set to the abstract value “Secured Dialogue”, the CF shall:

a) Retrieve the Local and Remote Entity IDs,

Note 1.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Local and Remote Entity IDs are retrieved is a local implementation matter.

b) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-10.

SA-SEND Request parameter	ATN value
Remote Entity ID	as specified in a) above
Local Entity ID	as specified in a) above
User Data	D-START User Data parameter

c) Store the D-START Request parameters for future use

Note 2.— The parameters will be required after the Security ASO finishes processing the SA-SEND Request by issuing an APDU to send via A-ASSOCIATE.

d) Submit the SA-SEND Request primitive to the Security ASO (see 4.8.3.3)

e) Enter the ASSOCIATION PENDING state as an initiator CF.

4.3.3.3.2.3 Quality of Service parameter mappings

Note.— The following paragraphs specify how the Quality of Service parameters in D-START Request and Response primitives are conveyed to the ATN Internet.

4.3.3.3.2.3.1 The Routing Class component of the quality of service parameter in D-START Request and Response primitives shall be conveyed to the ATN Internet and mapped to ATN Security Label by local means, using the values for Security Tag Value specified in Table 5.6-1.

Note.— 5.2.7.3.1 states that the mechanism by which the 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 Transport Service Access Point (TSAP), or be identified using a Systems Management function.

4.3.3.3.2.3.2 If no value for Routing Class is specified in the D-START Request primitive, then a default value shall be assigned as follows:

a) If the ATN-App AE is one of the ATS applications specified in 2.1 - 2.4, the value corresponding to “ATSC: No Traffic Type Policy Preference” is assigned;

b) otherwise, the traffic type defaults to General Communications, and no Security Tag Value is conveyed.

4.3.3.3.2.3.3 The Routing Class value conveyed to the ATN Internet when the D-START Response primitive is invoked shall be the same as that which was passed to the DS-User in the D-START Indication primitive.

4.3.3.3.2.3.4 The Priority component of the quality of service parameter in D-START Request and Response primitives shall be provided to the TS-Provider, by implementation-specific means, using the values for "Transport Layer Priority" specified in Table 1-2.

Note.— Although transport priority and network priority are semantically independent of each other, 5.5.1.2 requires that the Transport Service (TS)-user specifies the Application Service Priority, which in turn is mapped into the resulting Connectionless Network Protocol (CLNP) PDUs according to Table 1-2, which defines the fixed relationship between transport priority and the network priority.

4.3.3.3.2.3.5 If no value for Priority is specified in the D-START Request primitive, then the value corresponding to "Network/systems administration" shall be used.

4.3.3.3.2.3.6 The Priority value conveyed when the D-START Response primitive is invoked shall be the same as that which was passed to the DS-User in the D-START Indication primitive.

4.3.3.3.2.3.7 If the Routing Class parameter has an ATSC value, then the residual error rate (RER) component of the A-ASSOCIATE Quality of Service parameter shall be set to the logical value that maps to the lowest RER (highest integrity) supported by the transport service.

Note.— The A-ASSOCIATE residual error rate is mapped to the T-CONNECT RER parameter, the use of which is specified in 5.5.2.3.3.

4.3.3.3.2.3.8 If the Routing Class parameter has a non-ATSC value, then the residual error rate (RER) component of the quality of service parameter in D-START Request primitives shall map to the residual error rate component of the A-ASSOCIATE Quality of Service parameter.

4.3.3.3.2.3.9 The RER value in the D-START Response primitive shall be taken to be the same as that which was passed to the DS-User in the D-START Indication primitive.

Note.— RER is only present in the D-START response for backward compatibility with the previous version of the DS. It is not used in the current version.

4.3.3.3.3 D-START Response primitive

4.3.3.3.3.1 When Invoked

4.3.3.3.3.1.1 The D-START Response primitive may be validly invoked by the ATN-App ASE when the CF is the responder CF (see 4.3.3.6.1.2.1) and is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.3.3.2 Action Upon Invocation

4.3.3.3.3.2.1 When a D-START Response primitive is validly invoked on a dialogue for which security was not initially required, with the Security Requirements parameter absent, or set to the abstract value “No Security”, the CF shall :

a) Construct the Application Context name, with the value of the "version" arc set equal to the DS-User Version Number parameter if provided, and set to zero otherwise.

b) Retrieve the responding Presentation address

c) Make no use of the A-ASSOCIATE parameter “ACSE Requirements”.

d) Construct an A-ASSOCIATE Response primitive with the following parameters:

Table 4.3-11.

A-ASSOCIATE Response parameter	**ISO Status**	**ATN Value**
Application Context Name	M	As derived in a) above
Application Context Name List	C	Not used
Responding AP Title	U	Not used
Responding AE Qualifier	U	Not used
Responding AP Invocation-identifier	U	Not used
Responding AE Invocation-identifier	U	Not used
ACSE Requirements	C	As derived in c) above
Authentication-mechanism-name	U	Not used
Authentication-value	U	Not used
User Information	U	D-START User Data parameter

A-ASSOCIATE Response parameter	**ISO Status**	**ATN Value**
Result	M	D-START Result parameter
Diagnostic	U	Not used
Responding Presentation Address	M	Derived as in b) above
Presentation Context Definition Result List	C	Not used
Default Presentation Context Result	C	Not used
Quality of Service	M	As for D-START Request (see 4.3.3.3.2.3)
Presentation Requirements	U	Not used (default value)
Session Requirements	M	No Orderly Release (NOR), Duplex
Initial Synchronization Point Serial No	C	Not used
Initial Assignment of Tokens	C	Not used
Session-connection Identifier	U	Not used

e) invoke the A-ASSOCIATE Response, and

f) remain in the same state.

4.3.3.3.3.2.2 When the D-START Response is validly invoked on a dialogue supporting key management, with the Security Requirements parameter set to the abstract value “Secured Dialogue Supporting Key Management”, the CF shall:

a) Retrieve the called and calling Entity IDs,

Note.— The syntax of the SA-START parameters is described in 4.8. The way that the Calling AP Title and the Calling AE-Qualifier are retrieved is a local implementation matter.

b) Construct a SA-START Response primitive with the following parameters:

Table 4.3-12.

SA-START Response parameter	**ATN value**
Called Entity ID	as specified in a) above

SA-START Response parameter	**ATN value**
Calling Entity ID	as specified in a) above
User Data	D-START Response User Data parameter

c) Store the D-START Response parameters for future use

Note.— The parameters will be required after the Security ASO finishes processing the SA-START Request by issuing an APDU to send via A-ASSOCIATE.

d) Submit the SA-START Response primitive to the Security ASO (see 4.8.3.2)

e) Remain in the ASSOCIATION PENDING state.

4.3.3.3.3.2.3 When the D-START Response is validly invoked on a dialogue supporting a secured exchange, with the Security Requirements parameter set to the abstract value “Secured Dialogue”, the CF shall:

a) Retrieve the Local and Remote Entity IDs,

Note.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Local and Remote Entity IDs are retrieved is a local implementation matter.

b) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-13.

SA-SEND Request parameter	**ATN value**
Local Entity ID	as specified in a) above.
Remote Entity ID	as specified in a) above.
User Data	D-START Response user data.

c) Store the D-START Response parameters for future use

Note.— The parameters will be required after the Security ASO finishes processing the SA-SEND Request by issuing an APDU to send via A-ASSOCIATE.

d) Submit the SA-SEND Request primitive to the Security ASO (see 4.8.3.3)

e) Remain in the ASSOCIATION PENDING state.

4.3.3.3.4 D-END Request primitive

4.3.3.3.4.1 When Invoked

4.3.3.3.4.1.1 The D-END Request primitive may be validly invoked by the ATN-App ASE when the CF is in the DATA TRANSFER state; if it is in any other state then appropriate error recovery action shall be taken.

Note.— For example, if the CF is in the RELEASE PENDING state, then the D-END Request is rejected locally, with an appropriate result code.

4.3.3.3.4.2 Action Upon Invocation

4.3.3.3.4.2.1 When a D-END Request primitive is validly invoked on a dialogue which does not support security, the CF shall :

a) Construct an A-RELEASE Request primitive with the following parameter values:

Table 4.3-14.

A- RELEASE Request parameter	**ISO Status**	**ATN Value**
Reason	U	"normal"
User Information	U	D-END User Data parameter

b) Invoke the A-RELEASE Request primitive; and

c) Enter the RELEASE PENDING state as the Release Initiator CF.

4.3.3.3.4.2.2 When the D-END Request is validly invoked on a dialogue implementing support for key management or a secured exchange, the CF shall:

a) Retrieve the Calling and Called Peer IDs

Note.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Calling Peer ID and the Called Peer ID are retrieved is a local implementation matter.

b) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-15.

SA-SEND Request parameter	**ATN value**
Local Entity ID	as specified in a) above
Remote Entity ID	as specified in a) above
User Data	D-END Request User Data parameter

c) Submit the SA-SEND Request primitive to the Security ASO (see 4.8.3.3)

Note.— The Security ASO will subsequently process the SA-SEND Request and issue an APDU to send via A-RELEASE.

d) Enter the RELEASE PENDING state as the Release Initiator CF.

4.3.3.3.5 D-END Response primitive

4.3.3.3.5.1 When Invoked

4.3.3.3.5.1.1 The D-END Response primitive may be validly invoked by the ATN-App ASE when the CF is the Release Responder CF and is in the RELEASE PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.3.5.2 Action Upon Invocation

4.3.3.3.5.2.1 When a D-END Response primitive is validly invoked on a dialogue which does not support security and the *Result* parameter has the value “accepted”, the CF shall:

a) Construct an A-RELEASE Response primitive with parameter values as follows:

Table 4.3-16.

A- RELEASE Response parameter	**ISO Status**	**ATN Value**
Reason	U	“normal”
User Information	U	D-END User Data parameter
Result	M	“affirmative”

b) Invoke the A-RELEASE Response primitive

c) Remain in the RELEASE PENDING state.

4.3.3.3.5.2.2 When a D-END Response primitive is validly invoked on a dialogue which does not support security and the *Result* parameter has the abstract value “rejected” the CF shall:

a) Construct an A-RELEASE Response primitive with parameter values as follows:

Table 4.3-17.

A- RELEASE Response parameter	**ISO Status**	**ATN Value**
Reason	U	“not finished”
User Information	U	D-END User Data parameter
Result	M	“negative”

b) Invoke the A-RELEASE Response primitive; and

c) Remain in the RELEASE PENDING state.

4.3.3.3.5.2.3 When the D-END Response is validly invoked on a dialogue implementing support for key management or a secured exchange, the CF shall:

a) Retrieve the Calling and Called Peer IDs

Note.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Calling Peer ID and the Called Peer ID are retrieved is a local implementation matter.

b) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-18.

SA-SEND Request parameter	**ATN value**
Local Entity ID	as specified in a) to c) above
Remote Entity ID	as specified in a) to c) above
User Data	D-END User Data parameter

c) Store the D-END Response parameters for future use

Note.— The parameters will be required after the Security ASO finishes processing the SA-SEND Request by issuing an APDU to send via A-RELEASE.

d) Submit the SA-SEND Request primitive to the Security ASO (see 4.8.3.3)

e) Remain in the RELEASE PENDING state.

4.3.3.3.6 D-DATA Request primitive

4.3.3.3.6.1 When Invoked

4.3.3.3.6.1.1 The D-DATA Request primitive may be validly invoked by the ATN-App ASE when the CF is in the DATA TRANSFER state, or (if it is the Release Responder) in the RELEASE PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.3.6.2 Action Upon Invocation

4.3.3.3.6.2.1 When a D-DATA Request primitive is validly invoked on a dialogue which does not support security, the CF shall :

a) Using the definition of presentation-user-data in 4.3.2.6, encode the D-DATA Request User Data parameter with presentation-context-identifier value corresponding to “user-ase-apdu”;

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Remain in the same state.

4.3.3.3.6.2.2 When the D-DATA Request is validly invoked on a dialogue implementing support for key management or a secured exchange, the CF shall:

a) Retrieve the Local and Remote Entity IDs,

Note.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Local Entity ID and the Remote Entity ID are retrieved is a local implementation matter.

b) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-19.

SA-SEND Request parameter	ATN value
Local EntityID	as specified in a) above
Remote Entity ID	as specified in a) above
User Data	D-DATA User Data parameter

c) Submit the SA-SEND Request primitive to the Security ASO (see 4.8.3.3)

Note.— The Security ASO will subsequently process the SA-SEND Request and issue an APDU to send via P-DATA.

d) Remain in the same state.

4.3.3.3.7 D-ABORT Request primitive

4.3.3.3.7.1 When Invoked

4.3.3.3.7.1.1 Invocations of the D-ABORT Request primitive by the ATN-App ASE shall be allowed when the CF is in any valid state, except the NULL state; if an invocation occurs when the CF is in the NULL state then an error has occurred (see 4.3.3.1.2.4).

4.3.3.3.7.2 Action Upon Invocation

4.3.3.3.7.2.1 When a D-ABORT Request primitive is validly invoked on a dialogue which does not support security or on a dialogue implementing support for key management or a secured exchange in ASSOCIATION PENDING, RELEASE PENDING, or RELEASE COLLISION state, the CF shall:

a) If the Originator parameter of the D-ABORT has the symbolic value "User", then set Diagnostic to "No reason given". If the Originator parameter is absent or has any symbolic value other than "User", then set Diagnostic to "Protocol error".

b) Construct an A-ABORT Request primitive with the following parameter values:

Table 4.3-20.

A-ABORT Request parameter	ISO Status	ATN Value
Diagnostic	U	derived as in a) above

A-ABORT Request parameter	**ISO Status**	**ATN Value**
User Information	U	D-ABORT User Data parameter, if present and not empty.

c) Invoke the A-ABORT Request primitive; and

d) Remain in the same state.

4.3.3.3.7.2.2 When the D-ABORT Request is validly invoked on a dialogue implementing support for key management or a secured exchange in DATA TRANSFER state, the CF shall:

a) Retrieve the Local and Remote Entity IDs,

Note.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Local Entity ID and the Remote Entity ID are retrieved is a local implementation matter.

b) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-21.

SA-SEND Request parameter	**ATN value**
Local Entity ID	as specified in a) above
Remote Entity ID	as specified in a) above
User Data	D-ABORT User Data parameter

c) Store the D-ABORT Request parameters for future use

Note.— The parameters will be required after the Security ASO finishes processing the SA-SEND Request by issuing an APDU to send via A-ABORT.

d) Submit the SA-SEND Request primitive to the Security ASO (see 4.8.3.3)

e) Remain in the same state.

4.3.3.4 ACSE Services delivered to the CF

Note.— Events which occur at the upper service boundary of ACSE, i.e. Indication and Confirmation primitives which are generated by the ACPM and which require handling by the CF, are defined here.

4.3.3.4.1 A-ASSOCIATE Indication primitive

4.3.3.4.1.1 When Invoked

4.3.3.4.1.1.1 The A-ASSOCIATE Indication primitive may be validly invoked by the ACSE Protocol Machine (ACPM) when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.4.1.2 Action Upon Invocation

4.3.3.4.1.2.1 When an A-ASSOCIATE Indication primitive is validly invoked with the ACSE Requirements parameter not present, the CF shall:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Indication primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Indication.

b) If the Calling AP Title parameter is present, extract the Calling Peer Id from it. If the Calling AP Title contains an <app-type> arc, then extract the Calling Sys-ID from the Calling AE Qualifier parameter, if present. If the Calling AP Title parameter is not present, extract the Calling Presentation Address.

c) Construct a D-START Indication primitive, with the following parameter values:

Table 4.3-22.

D-START Indication parameter	**Value**
Calling Peer ID	Derived as in b) above
Calling Sys-ID	Derived as in b) above
Calling Presentation Address	Derived as in b) above
DS-User Version Number	Derived as in a) above
Security Requirements	"No Security"
Quality Of Service	See 4.3.3.4.1.3
User Data	A-ASSOCIATE User Information parameter

d) Invoke the D-START Indication primitive; and

e) Remain in the ASSOCIATION PENDING state.

4.3.3.4.1.2.2 When an A-ASSOCIATE Indication primitive is validly invoked with the ACSE Requirements and the Calling AP Title parameters both present, the CF shall:

Note 1.— The establishment of a Dialogue supporting either key management or a secured exchange mandates the presence of the Calling AP Title parameter in the A-ASSOCIATE Indication primitive.

a) Retrieve the <app-type> arc from the Calling AP Title parameter and check that it is the same as the <app-type> of the local DS-User,

Note 2.— The way the <app-type> of the local DS-User is know by the Dialogue CF is a local implementation matter.

Note 3.— If the <app-type> specified in the Calling AP Title parameter and the <app-type> of the local DS-User are not the same, then default local error handling procedures apply.

b) Retrieve the SESE PDU from the ACSE Authentication Value parameter,

c) Store the A-ASSOCIATE Indication parameters for future use

Note 4.— The parameters will be required after the Security ASO finishes processing the APDU by issuing an SA-service indication primitive.

d) Deliver the SESE PDU and the contents of the ACSE User Information parameter (which may be empty) to the Security ASO (see 4.8.5.2.2.9).

e) Remain in the ASSOCIATION PENDING state.

4.3.3.4.1.3 Quality of Service parameter mappings

Note.— The following paragraphs specify how the Quality of Service parameters in A-ASSOCIATE Indication and Confirmation primitives are conveyed to the DS-User as parameters of the D-START Indication and Confirmation primitives.

4.3.3.4.1.3.1 The Routing Class component of the quality of service parameter in D-START indication and confirmation primitives shall be obtained from the ATN Internet by local means, using the abstract values for Security Tag Values as specified in Table 5.6-1.

4.3.3.4.1.3.2 The Priority component of the quality of service parameter in D-START indication and confirmation primitives shall be taken from information provided by the TS-Provider, by implementation-specific means, using the abstract values for “Transport Layer Priority” specified in Table 1-2.

4.3.3.4.1.3.3 The RER component of the quality of service parameter in D-START indication and confirmation primitives shall be derived from the residual error rate component of the A-ASSOCIATE Quality of Service parameter as follows:

a) for ATSC applications, if the A-ASSOCIATE value equates to the highest available integrity level (as a minimum, the standard 16-bit transport checksum is required to be used) then set RER to the logical value "low"; otherwise set it to "high",

b) for non-ATSC applications, RER is mapped directly from the A-ASSOCIATE value.

4.3.3.4.2 A-ASSOCIATE Confirmation primitive

4.3.3.4.2.1 When Invoked

4.3.3.4.2.1.1 The A-ASSOCIATE Confirmation primitive may be validly invoked by the ACPM when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.4.2.2 Action Upon Invocation

4.3.3.4.2.2.1 When an A-ASSOCIATE Confirmation primitive is validly invoked on a dialogue for which no security was initially required, and the ACSE Requirements parameter is absent, and the *Result* parameter has the abstract value “accepted” the CF shall:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation.

b) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-23.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	“No Security”

D-START Confirmation parameter	**Value**
Quality Of Service	As for A-ASSOCIATE Indication (see 4.3.3.4.1.3)
Result	"accepted"
Reject Source	Not used
User Data	A-ASSOCIATE User Information parameter

c) Invoke the D-START Confirmation primitive

d) Enter the DATA TRANSFER state as the initiator CF.

4.3.3.4.2.2.2 When an A-ASSOCIATE Confirmation primitive is validly invoked and the ACSE Requirements parameter is absent, and the *Result* parameter has the abstract value "rejected (permanent)" or "rejected (transient)" the CF shall:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation primitive.

b) If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value "ACSE service-user" form a Reject Source parameter with value "DS user". If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value "ACSE service-provider" or "presentation service-provider" form a Reject Source parameter with value "DS provider".

c) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-24.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	"No Security"
Quality Of Service	As for A-ASSOCIATE Indication (see 4.3.3.4.1.3)
Result	"rejected (permanent)" or "rejected (transient)", from the A-ASSOCIATE Result parameter

D-START Confirmation parameter	**Value**
Reject Source	Derived as in b) above
User Data	A-ASSOCIATE User Information parameter

d) Invoke the D-START Confirmation primitive

e) Enter the NULL state.

4.3.3.4.2.2.3 When an A-ASSOCIATE Confirmation primitive is validly invoked on a dialogue for which security was initially required, with a valid ACSE Authentication value parameter, the CF shall:

a) Retrieve the SESE PDU from the ACSE Authentication value parameter,

b) Store the A-ASSOCIATE Confirmation parameters for future use,

Note.— The parameters will be required after the Security ASO finishes processing the APDU by issuing an SA-service indication primitive.

c) Deliver the SESE PDU and the contents of the ACSE User Information parameter (which may be empty) to the Security ASO (see 4.8.5.2.2.9),

d) Remain in the ASSOCIATION PENDING state as an initiator CF.

4.3.3.4.2.2.4 When an A-ASSOCIATE Confirmation primitive is validly invoked on a dialogue for which security was initially required, with the *ACSE Requirements* parameter not set, and the *Result* parameter has the abstract value "accepted", the CF shall:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation.

b) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-26.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	"No Security"

D-START Confirmation parameter	**Value**
Quality Of Service	As for A-ASSOCIATE Indication (see 4.3.3.4.1.3)
Result	"accepted"
Reject Source	Not used
User Data	A-ASSOCIATE User Information parameter

c) Invoke the D-START Confirmation primitive

d) Enter the DATA TRANSFER state as the initiator CF.

4.3.3.4.3 A-RELEASE Indication primitive

4.3.3.4.3.1 When Invoked

4.3.3.4.3.1.1 The A-RELEASE Indication primitive may be validly invoked by the ACPM when the CF is in the RELEASE PENDING or the RELEASE COLLISION state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.4.3.2 Action Upon Invocation

4.3.3.4.3.2.1 When an A-RELEASE Indication primitive is validly invoked on a dialogue which does not support security, and the CF is in the RELEASE PENDING state, it shall:

a) Construct a D-END Indication primitive, with the User Data parameter set equal to the value of the User Information parameter of the A-RELEASE Indication primitive.

b) Invoke the D-END Indication

c) Remain in the RELEASE PENDING state.

4.3.3.4.3.2.2 When an A-RELEASE Indication primitive is validly invoked on a dialogue which does not support security, and the CF is in the RELEASE COLLISION state, and it is the Initiator CF, it shall:

a) Construct a D-END Confirmation primitive, with the User Data parameter set equal to the value of the User Information parameter of the A-RELEASE Indication primitive, if present.

Note.— The D-END Confirmation is not issued to the DS-User until the orderly release procedure is complete, and an A-RELEASE Confirmation is received.

b) Construct an A-RELEASE response primitive with parameter values as follows:

Table 4.3-27.

A- RELEASE Response parameter	**ISO Status**	**ATN Value**
Reason	U	"normal"
User Information	U	Not present
Result	M	"affirmative"

c) Invoke the A-RELEASE Response primitive; and

d) Remain in the RELEASE COLLISION state.

4.3.3.4.3.2.3 When an A-RELEASE Indication primitive is validly invoked on a dialogue which does not support security, and the CF is in the RELEASE COLLISION state, and it is the Responder CF, it shall:

a) Construct a D-END Confirmation primitive, with the User Data parameter set equal to the value of the User Information parameter of the A-RELEASE Indication primitive, if present.

Note.— The D-END Confirmation is not issued to the DS-User until the orderly release procedure is complete, and an A-RELEASE Confirmation is received.

b) Remain in the RELEASE COLLISION state.

4.3.3.4.3.2.4 When the A-RELEASE Indication is validly invoked on a dialogue implementing support for key management or a secured exchange, the CF shall:

a) Retrieve the SESE PDU from the ACSE *User Information* parameter,

b) Deliver the APDU to the Security ASO (see 4.8.5.2.2.9).

c) Remain in the same state.

4.3.3.4.4 A-RELEASE Confirmation primitive

4.3.3.4.4.1 When Invoked

4.3.3.4.4.1.1 The A-RELEASE Confirmation primitive may be invoked by the ACPM when the CF is in the RELEASE PENDING or RELEASE COLLISION state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.4.4.2 Action Upon Invocation

4.3.3.4.4.2.1 When an A-RELEASE Confirmation primitive is validly invoked on a dialogue which does not support security, and the CF is in the RELEASE PENDING state, and the *Result* parameter has the abstract value "affirmative" the CF shall:

a) Construct a D-END Confirmation primitive with the following parameter values.

Table 4.3-28.

D-END Confirmation parameter	**Value**
Result	"affirmative"
User Data	User Information parameter from the A-RELEASE Confirmation, if present

b) Invoke the D-END Confirmation primitive.

c) Issue a P-U-ABORT request primitive, with no parameters.

Note.— This will cause the release of the underlying transport connection.

d) Enter the NULL state.

4.3.3.4.4.2.2 When an A-RELEASE Confirmation primitive is validly invoked on a dialogue which does not support security, and the CF is in the RELEASE PENDING state, and the *Result* parameter has the abstract value "negative" the CF shall:

a) Construct a D-END Confirmation primitive with the following parameter values.

Table 4.3-29.

D-END Confirmation parameter	**Value**
Result	“rejected”
User Data	User Information parameter from the A-RELEASE Confirmation, if present

b) Invoke the D-END Confirmation primitive.

c) Enter the DATA TRANSFER state.

4.3.3.4.4.2.3 When an A-RELEASE Confirmation primitive is validly invoked on a dialogue which does not support security, and the *Result* parameter has the abstract value “affirmative”, and the CF is in the RELEASE COLLISION state, and it is the Initiator CF, it shall:

a) Issue the D-END Confirmation primitive, which was previously formed in response to the reception of an A-RELEASE Indication primitive, to the DS-User.

b) Issue a P-U-ABORT request primitive, with no parameters.

Note.— This will cause the release of the underlying transport connection.

c) Enter the NULL state.

4.3.3.4.4.2.4 When an A-RELEASE Confirmation primitive is validly invoked on a dialogue which does not support security, and the *Result* parameter has the abstract value “affirmative”, and the CF is in the RELEASE COLLISION state, and it is the Responder CF, it shall:

a) Issue the D-END Confirmation primitive, which was previously formed in response to the reception of an A-RELEASE Indication primitive, to the DS-User.

b) Construct an A-RELEASE Response primitive, with the *Result* parameter set to “affirmative”.

c) Invoke the A-RELEASE Response.

d) Remain in the RELEASE COLLISION state.

4.3.3.4.4.2.5 When the A-RELEASE Confirmation is validly invoked on a dialogue implementing support for key management or a secured exchange, and the *Result* parameter has the abstract value “affirmative” the CF shall:

a) Retrieve the SESE PDU from the ACSE *User Information* parameter,

b) Store the A-RELEASE Confirmation parameters for future use,

Note.— The parameters will be required after the Security ASO finishes processing the APDU by issuing an SA-service indication primitive.

c) Deliver the APDU to the Security ASO (see 4.8.5.2.2.9),

d) Remain in the same state.

4.3.3.4.4.2.6 When the A-RELEASE Confirmation is validly invoked on a dialogue implementing support for key management or a secured exchange, and the CF is in the RELEASE PENDING state, and the *Result* parameter has the abstract value “negative” the CF shall:

a) Retrieve the SESE PDU from the ACSE *User Information* parameter,

b) Store the A-RELEASE Confirmation parameters for future use,

Note.— The parameters will be required after the Security ASO finishes processing the APDU by issuing an SA-service indication primitive.

c) Deliver the APDU to the Security ASO (see 4.8.5.2.2.9),

d) Remain in the same state.

4.3.3.4.5 A-ABORT Indication primitive

4.3.3.4.5.1 When Invoked

4.3.3.4.5.1.1 Invocations of the A-ABORT Indication primitive by the ACPM shall be allowed when the CF is in any valid state, except the NULL state; if an invocation occurs when the CF is in the NULL state then an error has occurred (see 4.3.3.1.2.4).

4.3.3.4.5.2 Action Upon Invocation

4.3.3.4.5.2.1 When an A-ABORT Indication primitive is validly invoked on a dialogue which does not support security, the CF shall:

a) If the *Abort Source* parameter of the A-ABORT Indication is set to "ACSE service-user" and the Diagnostic parameter is set to "No reason given", issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to "User" and the User Data parameter set equal to the User Information parameter in the A-ABORT Indication, if present.

b) If the *Abort Source* parameter of the A-ABORT Indication is set to "ACSE service-user" and the Diagnostic parameter is absent or is set to any value other than "No reason given", then issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to "Provider" and the User Data parameter set equal to the User Information parameter in the A-ABORT Indication, if present.

c) If the *Abort Source* parameter of the A-ABORT Indication has the abstract value "ACSE service-provider", then issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to the abstract value "Provider", and the User Data parameter set equal to the User Information parameter in the A-ABORT Indication, if present.

d) Enter the NULL state.

4.3.3.4.5.2.2 When the A-ABORT Indication is validly invoked on a dialogue implementing support for key management or a secured exchange, and the CF is in DATA TRANSFER state, the CF shall:

a) Retrieve the SESE PDU from the ACSE *User Information* parameter,

b) Store the A-ABORT Indication parameters for future use,

Note.— The parameters will be required after the Security ASO finishes processing the APDU by issuing an SA-service indication primitive.

c) Deliver the APDU to the Security ASO (see 4.8.5.2.2.9),

d) Remain in the same state.

4.3.3.4.5.2.3 When the A-ABORT Indication is validly invoked on a dialogue implementing support for key management or a secured exchange, and the CF is in the ASSOCIATION PENDING, RELEASE PENDING, or RELEASE COLLISION state, the CF shall:

a) If the *Abort Source* parameter of the A-ABORT Indication is set to "ACSE service-user" and the Diagnostic parameter is set to "No reason given", issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to "User" and the User Data parameter set equal to the User Information parameter in the A-ABORT Indication, if present.

b) If the *Abort Source* parameter of the A-ABORT Indication is set to "ACSE service-user" and the Diagnostic parameter is absent or is set to any value other than "No reason given", then issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to "Provider" and the User Data parameter set equal to the User Information parameter in the A-ABORT Indication, if present.

c) If the *Abort Source* parameter of the A-ABORT Indication has the abstract value "ACSE service-provider", then issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to the abstract value "Provider", and the User Data parameter set equal to the User Information parameter in the A-ABORT Indication, if present.

d) Enter the NULL state.

4.3.3.4.6 A-P-ABORT Indication primitive

4.3.3.4.6.1 When Invoked

4.3.3.4.6.1.1 Invocations of the A-P-ABORT Indication primitive by the ACPM shall be allowed when the CF is in any valid state, except the NULL state; if an invocation occurs when the CF is in the NULL state then an error has occurred (see 4.3.3.1.2.4).

4.3.3.4.6.2 Action Upon Invocation

4.3.3.4.6.2.1 When an A-P-ABORT Indication primitive is validly invoked, the CF shall:

a) issue a D-P-ABORT Indication primitive to the DS-User, and discard any Provider Reason parameter in the A-P-ABORT Indication; and

b) Enter the NULL state.

4.3.3.5 Services used by ACSE

Note.— ACSE, edition 2 mandates the mapping of ACSE APDUs to the underlying presentation service provider. However, when the efficient encoding options of Session and Presentation protocols are used, the full Presentation service is no longer available. Therefore, invocations of presentation service primitives by ACSE are "intercepted" by the CF and re-mapped to the "actual" presentation service as appropriate.

4.3.3.5.1 P-CONNECT Request primitive

4.3.3.5.1.1 When Invoked

4.3.3.5.1.1.1 The P-CONNECT Request primitive may be validly invoked by the ACPM when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.5.1.2 Action Upon Invocation

4.3.3.5.1.2.1 When a P-CONNECT Request primitive is validly invoked, the CF shall transparently invoke the equivalent presentation service primitive and remain in the same state.

4.3.3.5.2 P-CONNECT Response primitive

4.3.3.5.2.1 When Invoked

4.3.3.5.2.1.1 The P-CONNECT Response primitive may be validly invoked by the ACPM when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.5.2.2 Action Upon Invocation

4.3.3.5.2.2.1 When the P-CONNECT Response primitive is validly invoked, the CF shall:

a) transparently invoke the equivalent presentation service primitive.

b) If the P-CONNECT Response Result parameter has the abstract value “acceptance” then enter the DATA TRANSFER state, otherwise enter the NULL state.

4.3.3.5.3 P-U-ABORT Request primitive

4.3.3.5.3.1 When Invoked

4.3.3.5.3.1.1 Invocations of the P-U-ABORT Request primitive by the ACPM shall be allowed when the CF is in any valid state.

4.3.3.5.3.2 Action Upon Invocation

4.3.3.5.3.2.1 When a P-U-ABORT Request primitive is validly invoked, the CF shall:

a) If the P-U-ABORT request user data parameter is present, and the CF is in the DATA TRANSFER state:

   1) Encode the presentation user data as indicated in 4.3.2 with the P-U-ABORT user data parameter (an ABRT APDU) as the presentation data value and presentation context identifier value corresponding to "acse-apdu".

   2) Invoke a P-DATA Request primitive with the resulting encoding as User Data.

b) Otherwise, invoke a P-U-ABORT Request primitive with no parameters.

Note.— This will cause the underlying transport connection to be disconnected.

c) Enter the NULL state.

4.3.3.5.4 P-RELEASE Request primitive

Note.— ACSE, edition 2 mandates the mapping of A-RELEASE APDUs (RLRQ and RLRE) to the P-RELEASE service. However, when the efficient encoding options of Session and Presentation protocols are used, the Session No-Orderly Release (NOR) functional unit is selected, and no mapping for the P-RELEASE service is available. In order to provide an orderly release service, the CF re-maps invocations of the P-RELEASE service at the lower service boundary of ACSE to invocations of the P-DATA service, with the release APDUs transferred as user information.

4.3.3.5.4.1 When Invoked

4.3.3.5.4.1.1 The P-RELEASE Request primitive may be validly invoked by the ACPM when the CF is in the RELEASE PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.5.4.2 Action Upon Invocation

4.3.3.5.4.2.1 When a P-RELEASE Request primitive is validly invoked, the CF shall:

a) Encode the presentation user data as indicated in 4.3.2.6 with the P-RELEASE user data parameter (a RLRQ APDU) as the presentation data value and presentation context identifier corresponding to "acse-apdu".

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Remain in the RELEASE PENDING state.

4.3.3.5.5 P-RELEASE Response primitive

4.3.3.5.5.1 When Invoked

4.3.3.5.5.1.1 The P-RELEASE Response primitive may be validly invoked by the ACPM when the CF is in the RELEASE PENDING or RELEASE COLLISION state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.5.5.2 Action Upon Invocation

4.3.3.5.5.2.1 When a P-RELEASE Response primitive is validly invoked, and the CF is in the RELEASE PENDING state, and the *Result* parameter has the abstract value "affirmative" the CF shall:

a) encode the presentation user data as indicated in 4.3.2 with the P-RELEASE user data parameter (a RLRE APDU) as the presentation data value and presentation context identifier corresponding to "acse-apdu".

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data.

c) Enter the NULL state.

Note.— The peer AEI is now expected to issue a P-U-ABORT request, which will cause the release of the underlying connection.

4.3.3.5.5.2.2 When a P-RELEASE Response primitive is validly invoked, and the CF is in the RELEASE PENDING state, and the *Result* parameter has the abstract value "negative" the CF shall:

a) Encode the presentation user data as indicated in 4.3.2 with the P-RELEASE user data parameter (a RLRE APDU) as the presentation data value and presentation context identifier corresponding to "acse-apdu";

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Enter the DATA TRANSFER state.

4.3.3.5.5.2.3 When a P-RELEASE Response primitive is validly invoked, and the CF is in the RELEASE COLLISION state, and it is the Initiator CF, it shall:

a) Encode the presentation user data as indicated in 4.3.2 with the P-RELEASE user data parameter (a RLRE APDU) as the presentation data value and presentation context identifier corresponding to "acse-apdu";

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Remain in the RELEASE COLLISION state.

4.3.3.5.5.2.4 When a P-RELEASE Response primitive is validly invoked, and the CF is in the RELEASE COLLISION state, and it is the Responder CF, it shall:

a) Encode the presentation user data as indicated in 4.3.2 with the P-RELEASE user data parameter (a RLRE APDU) as the presentation data value and presentation context identifier corresponding to "acse-apdu";

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Enter the NULL state.

Note.— The peer AEI is now expected to issue a P-U-ABORT request, which will cause the release of the underlying connection.

4.3.3.5.5.2.5 **Recommendation. -** *After entering the NULL state, implementations should release the underlying connection (e.g., by issuing the P-U-ABORT request) if the communication peer does not cause the connection to be released as expected, after a period of time not less than twice the anticipated end-to-end transit time.*

4.3.3.6 Supporting Services delivered to the CF

Note 1.— The mapping by the CF of presentation service indication and confirmation primitives, which are invoked by the presentation service provider, is defined in the following paragraphs.

Note 2.— The following provisions describe the behaviour to be exhibited by the ATN-App AE when the supporting communications service exhibits behaviour modelled by the passing of indication or confirmation primitives to the application layer.

4.3.3.6.1 P-CONNECT Indication primitive

4.3.3.6.1.1 When Invoked

4.3.3.6.1.1.1 When the P-CONNECT Indication primitive is invoked by the supporting service, a new instance of communication shall be created, with its CF initially in the NULL state.

4.3.3.6.1.2 Action Upon Invocation

4.3.3.6.1.2.1 When a P-CONNECT Indication primitive is validly invoked, the CF shall:

a) transparently invoke the equivalent presentation service primitive at the lower ACSE service boundary; and

b) enter the ASSOCIATION PENDING state as the responder CF.

4.3.3.6.2 P-CONNECT Confirmation primitive

4.3.3.6.2.1 When Invoked

4.3.3.6.2.1.1 The P-CONNECT Confirmation primitive may be validly invoked by the supporting service when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.6.2.2 Action Upon Invocation

4.3.3.6.2.2.1 When a P-CONNECT Confirmation primitive is validly invoked, the CF shall :

a) transparently invoke the equivalent presentation service primitive at the lower ACSE service boundary; and

b) Remain in the ASSOCIATION PENDING state.

4.3.3.6.3 P-U-ABORT Indication primitive

4.3.3.6.3.1 When Invoked

4.3.3.6.3.1.1 Invocations of the P-U-ABORT Indication primitive by the supporting service shall be allowed when the CF is in any valid state.

4.3.3.6.3.2 Action Upon Invocation

4.3.3.6.3.2.1 When a P-U-ABORT Indication primitive is validly invoked, the CF shall:

a) if the CF is in the NULL state, take no action;

b) otherwise, transparently invoke the equivalent presentation service primitive at the lower ACSE service boundary, and remain in the same state.

4.3.3.6.4 P-P-ABORT Indication primitive

4.3.3.6.4.1 When Invoked

4.3.3.6.4.1.1 Invocations of the P-P-ABORT Indication primitive by the supporting service shall be allowed when the CF is in any valid state.

4.3.3.6.4.2 Action Upon Invocation

4.3.3.6.4.2.1 When a P-P-ABORT Indication primitive is validly invoked, the CF shall:

a) if the CF is in the NULL state, then take no action;

b) otherwise, transparently invoke the corresponding presentation service primitive at the lower ACSE service boundary; and

c) remain in the same state.

4.3.3.6.5 P-DATA Indication primitive

4.3.3.6.5.1 When Invoked

4.3.3.6.5.1.1 Invocations of the P-DATA Indication primitive by the supporting service shall be allowed when the CF is in a valid state to receive the decoded APDU, as listed in 4.3.3.6.5.2; if an invocation occurs when the CF is not in a valid state then an error has occurred (see 4.3.3.1.2.4).

4.3.3.6.5.2 Action Upon Invocation

4.3.3.6.5.2.1 When a P-DATA Indication primitive is validly invoked, the CF shall decode the presentation user data as indicated in 4.3.2 to determine the destination ASE of the APDU, and extract the presentation data value.

Note.— The destination ASE is determined from the value of the presentation-context-identifier in the received User-data. Valid values are acse-apdu and user-ase-apdu, which correspond to destination ASEs of ACSE and ATN-App ASE, respectively.

4.3.3.6.5.2.2 ACSE APDU Received

4.3.3.6.5.2.2.1 If the destination ASE is ACSE then the CF shall determine the type of ACSE APDU present in the extracted presentation data value.

Note.— ACSE APDUs which may validly be received in a P-DATA indication are A-Release-Request (RLRQ), A-Release-Response (RLRE), and A-Abort (ABRT) APDUs.

4.3.3.6.5.2.2.2 If the received APDU is RLRQ, the CF shall:

a) if in the DATA TRANSFER state, then invoke a P-RELEASE Indication primitive at the ACSE lower service boundary, with the RLRQ as User Data, and enter the RELEASE PENDING state as the Release Responder CF;

b) if in the RELEASE PENDING state, and the CF is the Release Initiator, then invoke a P-RELEASE Indication primitive at the ACSE lower service boundary with the RLRQ as User Data, and enter the RELEASE COLLISION state;

c) if in the NULL state, and this CF has previously issued an ABRT APDU and is awaiting disconnection by the peer, then take no action and remain in the NULL state;

d) if none of the conditions a) to c) is satisfied, then take error handling action as described in 4.3.3.6.5.2.4.

4.3.3.6.5.2.2.3 If the received APDU is RLRE, the CF shall:

a) if the Reason field in the RLRE has the value "not-finished", and the CF is in the RELEASE PENDING state, then invoke a P-RELEASE Confirmation primitive at the ACSE lower service boundary, with the result parameter set to "negative", and the RLRE as User Data; remain in the RELEASE PENDING state;

b) if the Reason field in the RLRE has the value "normal", and the CF is in the RELEASE PENDING or RELEASE COLLISION state, then invoke a P-RELEASE Confirmation primitive at the ACSE lower service boundary, with the result parameter set to "affirmative", and the RLRE as User Data; remain in the same state;

c) if the CF is in the NULL state, and this CF has previously issued an ABRT APDU and is awaiting disconnection by the peer, then take no action and remain in the NULL state;

d) if none of the conditions a) to c) is satisfied, then take error handling action in 4.3.3.6.5.2.4.

4.3.3.6.5.2.2.4 If the received APDU is ABRT, the CF shall:

a) if the CF is in the state DATA TRANSFER, or RELEASE PENDING, or RELEASE COLLISION, then invoke a P-U-ABORT Indication primitive at the ACSE lower service boundary, with the ABRT as User Data, and issue a P-U-ABORT request with no parameters to the underlying service; remain in the same state;

b) if the CF is in the NULL state, then take no action unless this CF has previously issued an ABRT APDU and is awaiting disconnection by the peer, in which case issue a P-U-ABORT request to the underlying service; remain in the same state;

c) if neither of the conditions a) and b) is satisfied, then take error handling action as described in 4.3.3.6.5.2.4.

4.3.3.6.5.2.3 ATN-App APDU Received

4.3.3.6.5.2.3.1 If the destination ASE is ATN-App ASE and the dialogue does not support security, then the CF shall:

a) if the CF is in the DATA TRANSFER state, or the CF is in the RELEASE PENDING state and is the Release Initiator CF, then issue a D-DATA Indication primitive to the DS-User, with the received presentation data value as the user data parameter, and remain in the same state;

b) if the CF is in the NULL state, and this CF has previously issued an ABRT APDU and is awaiting disconnection by the peer, then take no action and remain in the same state;

c) if neither of the conditions a) and b) is satisfied, then take error handling action as described in 4.3.3.6.5.2.4.

4.3.3.6.5.2.3.2 If the destination ASE is ATN-App ASE and the dialogue implements support for key management or a secured exchange, then the CF shall:

a) if the CF is in the DATA TRANSFER state, or the CF is in the RELEASE PENDING state and is the Release Initiator CF, then deliver the received presentation data value to the Security ASO (see 4.8.5.2.2.9), and remain in the same state;

b) if the CF is in the NULL state, and this CF has previously issued an ABRT APDU and is awaiting disconnection by the peer, then take no action and remain in the same state;

c) if neither of the conditions a) and b) is satisfied, then take error handling action as described in 4.3.3.6.5.2.4.

4.3.3.6.5.2.4 Error conditions

4.3.3.6.5.2.4.1 If the destination ASE is invalid (i.e. neither ACSE nor ATN-App ASE), or an unrecognised APDU is received, or a valid APDU is received when the CF is not in the correct state (as defined in 4.3.3.6.5.2.2 and 4.3.3.6.5.2.3), then the CF shall:

a) if not in the NULL state then issue a P-U-ABORT request with no parameters to the supporting service: and

b) regardless of CF state behave as if a P-U-ABORT indication had been received.

4.3.3.7 Security ASO services delivered to the CF

4.3.3.7.1 (Paragraph deleted)

4.3.3.7.2 SA-START Indication primitive

4.3.3.7.2.1 When Invoked

4.3.3.7.2.1.1 The SA-START Indication primitive may be validly invoked by the S-ASO on a dialogue when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.7.2.2 Action Upon Invocation

4.3.3.7.2.2.1 When an SA-START Indication primitive is validly invoked, the CF shall retrieve the previously stored A-ASSOCIATE Indication parameters and:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Indication primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Indication.

b) Extract the Calling Peer Id from the Calling AP Title parameter. Extract the Calling Sys-ID from the Calling AE Qualifier parameter, if present.

c) Construct a D-START Indication primitive, with the following parameter values:

Table 4.3-36.

D-START Indication parameter	**Value**
Calling Peer ID	Derived as in b) above
Calling Sys-ID	Derived as in b) above
Calling Presentation Address	Not used
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue Supporting Key Management”
Quality Of Service	See 4.3.3.4.1.3.

D-START Indication parameter	**Value**
User Data	A-ASSOCIATE User Information parameter

d) Invoke the D-START Indication primitive; and

e) Remain in the ASSOCIATION PENDING state.

4.3.3.7.3 SA-START Confirmation primitive

4.3.3.7.3.1 When Invoked

4.3.3.7.3.1.1 The SA-START Confirmation primitive may be validly invoked by the S-ASO on a dialogue implementing support for key management, when the CF is in the ASSOCIATION PENDING state; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.7.3.2 Action Upon Invocation

4.3.3.7.3.2.1 When an SA-START Confirmation primitive is validly invoked, the CF shall retrieve the previously stored A-ASSOCIATE Confirmation parameters and, if the A-ASSOCIATE *Result* parameter has the abstract value “accepted”:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation.

b) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-37.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue Supporting Key Management”
Quality Of Service	See 4.3.3.4.1.3.
Result	“accepted”
Reject Source	Not used
User Data	A-ASSOCIATE User Information parameter

c) Invoke the D-START Confirmation primitive

d) Enter the DATA TRANSFER state as the initiator CF.

4.3.3.7.3.2.2 When an SA-START Confirmation primitive is validly invoked, the CF shall retrieve the previously stored A-ASSOCIATE Confirmation parameters and, if the A-ASSOCIATE *Result* parameter has the abstract value “rejected (permanent)” or “rejected (transient)”:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation primitive.

b) If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value “ACSE service-user,” form a Reject Source parameter with value “DS user”. If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value “ACSE service-provider” or “presentation service-provider,” form a Reject Source parameter with value “DS provider”.

c) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-38.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue Supporting Key Management”
Quality Of Service	See 4.3.3.4.1.3.
Result	“rejected (permanent)” or “rejected (transient)”, from the A-ASSOCIATE Result parameter
Reject Source	Derived as in b) above
User Data	A-ASSOCIATE User Information parameter

d) Invoke the D-START Confirmation primitive

e) Enter the NULL state.

4.3.3.7.4 SA-SEND Indication primitive

4.3.3.7.4.1 When Invoked

4.3.3.7.4.1.1 The SA-SEND Indication primitive may be validly invoked by the S-ASO on a dialogue implementing support for key management or a secured exchange, when the CF is in any of the ASSOCIATION PENDING, DATA TRANSFER, RELEASE PENDING, or RELEASE COLLISION states; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.7.4.2 Action Upon Invocation

4.3.3.7.4.2.1 When an SA-SEND Indication primitive is validly invoked, the CF is the initiator CF, and is in the ASSOCIATION PENDING state, the CF shall retrieve the previously stored A-ASSOCIATE Confirmation parameters and, if the A-ASSOCIATE *Result* parameter has the abstract value “accepted”:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation.

b) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-39.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue”
Quality Of Service	See 4.3.3.4.1.3
Result	“accepted”
Reject Source	Not used
User Data	SA-SEND indication User data parameter

c) Invoke the D-START Confirmation primitive

d) Enter the DATA TRANSFER state as the initiator CF.

4.3.3.7.4.2.2 When an SA-SEND Indication primitive is validly invoked, the CF is the initiator CF, and is in the ASSOCIATION PENDING state, the CF shall retrieve the previously stored A-ASSOCIATE Confirmation parameters and, if the A-ASSOCIATE *Result* parameter has the abstract value “rejected (permanent)” or “rejected (transient)”:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation primitive.

b) If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value “ACSE service-user,” form a Reject Source parameter with value “DS user”. If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value “ACSE service-provider” or “presentation service-provider,” form a Reject Source parameter with value “DS provider”.

c) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-41.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue”
Quality Of Service	See 4.3.3.4.1.3.
Result	“rejected (permanent)” or “rejected (transient)”, from the A-ASSOCIATE Result parameter
Reject Source	Derived as in b) above
User Data	SA-SEND indication User data parameter

d) Invoke the D-START Confirmation primitive

e) Enter the NULL state.

4.3.3.7.4.2.3 When an SA-SEND Indication primitive is validly invoked, the CF is the responder CF, and is in the ASSOCIATION PENDING state, the CF shall retrieve the previously stored A-ASSOCIATE Indication parameters and:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Indication primitive.

If it has the value zero, then omit the DS-User Version Number parameter in the D-START Indication.

b) If the Calling AP Title parameter is present, extract the Calling Peer Id from it. If the Calling AP Title contains an <app-type> arc, then extract the Calling Sys-ID from the Calling AE Qualifier parameter, if present. If the Calling AP Title parameter is not present, extract the Calling Presentation Address.

c) Construct a D-START Indication primitive, with the following parameter values:

Table 4.3-42.

D-START Indication parameter	**Value**
Calling Peer ID	Derived as in b) above
Calling Sys-ID	Derived as in b) above
Calling Presentation Address	Derived as in b) above
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue”
Quality Of Service	See 4.3.3.4.1.3.
User Data	SA-SEND Indication User Data parameter.

d) Invoke the D-START Indication primitive; and

e) Remain in the ASSOCIATION PENDING state.

4.3.3.7.4.2.4 When an SA-SEND Indication primitive is validly invoked, the CF is the initiator CF, and is in the ASSOCIATION PENDING state, the CF shall retrieve the previously stored A-ASSOCIATE Confirmation parameters and, if the A-ASSOCIATE *Result* parameter has the abstract value “rejected (permanent)” or “rejected (transient)”:

a) If the "version" component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-START Confirmation primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-START Confirmation primitive.

b) If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value “ACSE service-user,” form a Reject Source parameter with value “DS user”. If the A-ASSOCIATE Confirmation *Result Source* parameter has the abstract value “ACSE service-provider” or “presentation service-provider,” form a Reject Source parameter with value “DS provider”.

c) Construct a D-START Confirmation primitive, with parameter values as follows:

Table 4.3-44.

D-START Confirmation parameter	**Value**
DS-User Version Number	Derived as in a) above
Security Requirements	Abstract value “Secured Dialogue”
Quality Of Service	See 4.3.3.4.1.3.
Result	“rejected (permanent)” or “rejected (transient)”, from the A-ASSOCIATE Result parameter
Reject Source	Derived as in b) above
User Data	SA-SEND Indication User data parameter

d) Invoke the D-START Confirmation primitive

e) Enter the NULL state.

4.3.3.7.4.2.5 When an SA-SEND Indication primitive is validly invoked, the CF is in the DATA TRANSFER state, and is not aborting, the CF shall issue a D-DATA Indication primitive to the DS-User, with the SA-SEND Indication user data parameter as the user data parameter, and remain in the same state.

4.3.3.7.4.2.6 When an SA-SEND Indication primitive is validly invoked, the CF is in the DATA TRANSFER state, and the dialogue is aborting, the CF shall retrieve the previously stored A-ABORT Indication parameters and:

a) If the *Abort Source* parameter of the A-ABORT Indication is set to “ACSE service-user” and the Diagnostic parameter is set to “No reason given”, issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to “User” and the User Data parameter set equal to the SA-SEND Indication user data parameter, if present.

b) If the *Abort Source* parameter of the A-ABORT Indication is set to “ACSE service-user” and the Diagnostic parameter is absent or is set to any value other than “No reason given”, then issue a D-ABORT Indication primitive to the DS-User, with the Originator parameter set to “Provider” and the User Data parameter set equal to the SA-SEND Indication user data parameter, if present.

c) If the *Abort Source* parameter of the A-ABORT Indication has the abstract value “ACSE service-provider”, then issue a D-ABORT Indication primitive to the

DS-User, with the Originator parameter set to the abstract value “Provider”, and the User Data parameter set equal to the SA-SEND Indication user data parameter, if present.

d) Enter the NULL state.

Note.— The Dialogue is aborting when a D-ABORT Request or an A-ABORT Indication primitive has been locally issued.

4.3.3.7.4.2.7 When an SA-SEND Indication primitive is validly invoked, the CF is the release responder in the RELEASE PENDING state, the CF shall:

a) Construct a D-END Indication primitive, with the User Data parameter set equal to the value of the User Data parameter of the SA-SEND Indication primitive,

b) Invoke the D-END Indication,

c) Remain in the RELEASE PENDING state.

4.3.3.7.4.2.8 When an SA-SEND Indication primitive is validly invoked, the CF is the release initiator and is in the RELEASE PENDING state, the CF shall retrieve the previously stored A-RELEASE Confirmation parameters and, if the A-RELEASE *Result* parameter has the abstract value “affirmative”:

a) Construct a D-END Confirmation primitive with the following parameter values.

Table 4.3-47.

D-END Confirmation parameter	**Value**
Result	“accepted”
User Data	User Data parameter from the SA-SEND Indication, if present

b) Invoke the D-END Confirmation primitive.

c) Issue a P-U-ABORT request primitive, with no parameters.

Note.— This will cause the release of the underlying transport connection.

d) Enter the NULL state.

4.3.3.7.4.2.9 When an SA-SEND Indication primitive is validly invoked, the CF is the release initiator and is in the RELEASE PENDING state, the CF shall retrieve the previously stored A-RELEASE Confirmation parameters and, if the A-RELEASE *Result* parameter has the abstract value "negative":

a) Construct a D-END Confirmation primitive with the following parameter values.

Table 4.3-48.

D-END Confirmation parameter	**Value**
Result	"rejected"
User Data	User Data parameter from the SA-SEND Indication, if present

b) Invoke the D-END Confirmation primitive.

c) Enter the DATA TRANSFER state.

4.3.3.7.4.2.10 When an SA-SEND Indication primitive is validly invoked, the CF is the release initiator in the RELEASE PENDING state, and no A-RELEASE primitive was received, the CF shall issue a D-DATA Indication primitive to the DS-User, with the received data value as the user data parameter, and remain in the same state.

4.3.3.7.4.2.11 When an SA-SEND Indication primitive is validly invoked, the CF is the Initiator CF and is in the RELEASE COLLISION state, the CF shall retrieve the previously stored A-RELEASE Confirmation parameters and, if the A-RELEASE *Result* parameter has the abstract value "affirmative":

a) Issue the D-END Confirmation primitive, which was previously formed in response to the reception of an A-RELEASE Indication primitive, to the DS-User,

b) Issue a P-U-ABORT request primitive, with no parameters,

Note.— This will cause the release of the underlying transport connection.

c) Enter the NULL state.

4.3.3.7.4.2.12 When an SA-SEND Indication primitive is validly invoked, the CF is the Responder CF and is in the RELEASE COLLISION state, the CF shall retrieve the previously stored A-RELEASE Confirmation parameters and, if the A-RELEASE *Result* parameter has the abstract value "affirmative":

a) Issue the D-END Confirmation primitive, which was previously formed in response to the reception of an A-RELEASE Indication primitive, to the DS-User.

b) Construct an A-RELEASE Response primitive, with the *Result* parameter set to “affirmative”.

c) Invoke the A-RELEASE Response.

d) Remain in the RELEASE COLLISION state.

4.3.3.7.4.2.13 When an SA-SEND Indication primitive is validly invoked, and the CF is in the RELEASE COLLISION state, and it is the Initiator CF, and no A-RELEASE primitive was received, it shall:

a) Construct a D-END Confirmation primitive, with the User Data parameter set equal to the value of the Data parameter of the SA-SEND Indication primitive, if present.

Note.— The D-END Confirmation is not issued to the DS-User until the orderly release procedure is complete, and an A-RELEASE Confirmation is received.

b) Retrieve the Local and Remote Entity IDs,

Note.— The syntax of the SA-SEND parameters is described in 4.8. The way that the Local Entity ID and the Remote Entity ID are retrieved is a local implementation matter.

c) Construct a SA-SEND Request primitive with the following parameters:

Table 4.3-50.

SA-SEND Request parameter	**ATN value**
Local Entity ID	as specified in b) above
Remote Entity ID	as specified in b) above
User Data	Not used

d) Invoke the SA-SEND Request primitive

e) Remain in the RELEASE COLLISION state.

4.3.3.7.4.2.14 When an SA-SEND Indication primitive is validly invoked, and the CF is in the RELEASE COLLISION state, and it is the Responder CF, and no A-RELEASE primitive was received, it shall:

a) Construct a D-END Confirmation primitive, with the User Data parameter set equal to the value of the User Data of the SA-SEND Indication primitive, if present.

Note.— The D-END Confirmation is not issued to the DS-User until the orderly release procedure is complete, and an A-RELEASE Confirmation is received.

b) Remain in the RELEASE COLLISION state.

4.3.3.8 APDUs emitted by the Security ASO

4.3.3.8.1 APDU received from Security ASO

4.3.3.8.1.1 When Submitted

4.3.3.8.1.1.1 An APDU generated within the Security ASO may validly be submitted to the CF on a dialogue implementing support for key management or a secured exchange, when the CF is in any of the NULL, ASSOCIATION PENDING, DATA TRANSFER, RELEASE COLLISION or RELEASE PENDING states; if it is in any other state then appropriate error recovery action shall be taken.

4.3.3.8.1.2 Action Upon Receiving an APDU

4.3.3.8.1.2.1 When an APDU is validly submitted by the Security ASO, and the CF is in NULL state, the CF shall take no action.

4.3.3.8.1.2.2 When an APDU is validly submitted by the Security ASO, and the CF is in the ASSOCIATION PENDING state, and it is the Initiator CF, the CF shall retrieve the previously stored D-START Request parameters and:

a) Determine the app-type as defined for the ATN-App AE,

b) Construct the Application Context name, with the value of the "version" arc set equal to the DS-User Version Number parameter if it was provided in the D-START request, and set to zero otherwise,

c) If not specified in the D-START request primitive, retrieve the local Calling Presentation Address,

d) Determine the Called Presentation Address either directly from the D-START Called Presentation Address parameter if present, or via look-up from the Called Peer ID and Called Sys-ID parameters,

e) Construct the Calling AP Title from local knowledge of the <end-system-id> and <app-type>, as specified in 4.3.2.2.3. If this is a ground system with no registered ICAO facility designator, then set the first four characters of <end-system-id> to the reserved value “0000" (four zero characters). If the optional D-START parameter

Calling Sys-ID is present, then use this as the Calling AE-Qualifier. If Calling Sys-ID is not present, then Calling AE-Qualifier is not used in the A-ASSOCIATE request (and it will not then be included in the resulting A-ASSOCIATE-REQUEST (AARQ) APDU),

Note.— The way that the local end-system-id and app-type values are known is a local implementation matter. Any Calling Peer Id value that was provided by the DS-User in the D-START primitive is overridden.

f) If the dialogue is supporting key management, then retrieve the User Data parameter of the D-START Request and use this as the A-ASSOCIATE User Information parameter. If the dialogue is supporting a secured exchange then the User Data is already included in the SESE PDU and the A-ASSOCIATE User Information parameter is not used.

g) Construct an A-ASSOCIATE Request primitive with the following parameters:

Table 4.3-51.

A-ASSOCIATE Request parameter	**ISO Status**	**ATN value**
Mode	U	Not used (default value)
Application Context Name	M	As derived in b) above
Application Context Name List	C	Not used
Calling AP Title	U	As derived in e) above
Calling AE Qualifier	U	As derived in e) above
Calling AP Invocation-identifier	U	Not used
Calling AE Invocation-identifier	U	Not used
Called AP Title	U	Not used
Called AE Qualifier	U	Not used
Called AP Invocation-identifier	U	Not used
Called AE Invocation-identifier	U	Not used
ACSE Requirements	U	symbolic value “authentication”
Authentication-mechanism-name	U	Not used
Authentication-value	U	APDU received from the Security ASO

A-ASSOCIATE Request parameter	**ISO Status**	**ATN value**
User Information	U	As derived in f) above
Calling Presentation Address	M	Derived as in c) above
Called Presentation Address	M	Derived as in d) above
Presentation Context Definition List	U	Not used
Default Presentation Context Name	U	Not used
Quality of Service	M	See 4.3.3.3.2.3
Presentation Requirements	U	Not used (default value)
Session Requirements	M	No Orderly Release (NOR), Duplex
Initial Synchronization Point Serial No	C	Not used
Initial Assignment of Tokens	C	Not used
Session-connection Identifier	U	Not used

h) Invoke the A-ASSOCIATE Request primitive

i) Remain in the ASSOCIATION PENDING state.

4.3.3.8.1.2.3 When an APDU is validly submitted by the Security ASO, the CF is in the ASSOCIATION PENDING state, and it is the Responder CF, the CF shall retrieve the previously stored D-START Response parameters and:

a) Construct the Application Context name, with the value of the "version" arc set equal to the DS-User Version Number parameter if provided, and set to zero otherwise,

b) Retrieve the responding Presentation Address,

c) If the dialogue is supporting key management, then retrieve the User Data parameter of the D-START primitive and use this as the A-ASSOCIATE User Information parameter. If the dialogue is supporting a secured exchange then the User Data is already included in the SESE PDU and the A-ASSOCIATE User Information parameter is not used.

d) Construct an A-ASSOCIATE Response primitive with the following parameters:

Table 4.3-52.

A-ASSOCIATE Response parameter	**ISO Status**	**ATN Value**
Application Context Name	M	As derived in a) above
Application Context Name List	C	Not used
Responding AP Title	U	Not used
Responding AE Qualifier	U	Not used
Responding AP Invocation-identifier	U	Not used
Responding AE Invocation-identifier	U	Not used
ACSE Requirements	C	symbolic value “authentication”
Authentication-mechanism-name	U	Not used
Authentication-value	U	APDU received from the Security ASO
User Information	U	As derived in c) above
Result	M	D-START Result parameter
Diagnostic	U	Not used
Responding Presentation Address	M	Derived as in b) above
Presentation Context Definition Result List	C	Not used
Default Presentation Context Result	C	Not used
Quality of Service	M	See 4.3.3.3.2.3
Presentation Requirements	U	Not used (default value)
Session Requirements	M	No Orderly Release (NOR), Duplex
Initial Synchronization Point Serial No	C	Not used
Initial Assignment of Tokens	C	Not used
Session-connection Identifier	U	Not used

e) invoke the A-ASSOCIATE Response, and

f) remain in the same state.

4.3.3.8.1.2.4 When an APDU is validly submitted by the Security ASO, the CF is in the DATA TRANSFER state, and the dialogue is aborting, the CF shall retrieve the previously stored D-ABORT Request parameters and:

a) If the Originator parameter of the D-ABORT has the symbolic value "User", then set Diagnostic to "No reason given". If the Originator parameter is absent or has any symbolic value other than "User", then set Diagnostic to "Protocol error",

b) Construct an A-ABORT Request primitive with the following parameter values:

Table 4.3-53.

A-ABORT Request parameter	**ISO Status**	**ATN Value**
Diagnostic	U	derived as in a) above
User Information	U	APDU received from the Security ASO

c) Invoke the A-ABORT Request primitive; and

d) Remain in the same state.

4.3.3.8.1.2.5 When an APDU is validly submitted by the Security ASO, the CF is in the DATA TRANSFER state, and the dialogue is not aborting, the CF shall:

a) Using the definition of presentation-user-data in 4.3.2.6, encode the APDU received from the Security ASO with presentation-context-identifier value corresponding to "user-ase-apdu";

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Remain in the same state.

4.3.3.8.1.2.6 When an APDU is validly submitted by the Security ASO, the CF is the Release Initiator CF, it is in the RELEASE PENDING state, and the dialogue is not aborting, the CF shall:

a) Construct an A-RELEASE Request primitive with the following parameter values:

Table 4.3-54.

A- RELEASE Request parameter	ISO Status	ATN Value
Reason	U	"normal"
User Information	U	APDU received from the Security ASO

b) Invoke the A-RELEASE Request primitive; and

c) Remain in the same state.

4.3.3.8.1.2.7 When an APDU is validly submitted by the Security ASO, the CF is the Release Responder CF, and is in the RELEASE PENDING state, the CF shall retrieve the previously stored D-END Response parameters and, if the D-END *Result* parameter has the abstract value "accepted":

a) Construct an A-RELEASE Response primitive with parameter values as follows:

Table 4.3-55.

A- RELEASE Response parameter	ISO Status	ATN Value
Reason	U	"normal"
User Information	U	APDU received from the Security ASO
Result	M	"affirmative"

b) Invoke the A-RELEASE Response primitive,

c) Remain in the RELEASE PENDING state.

4.3.3.8.1.2.8 When an APDU is validly submitted by the Security ASO, the CF is the Release Responder CF, and is in the RELEASE PENDING state, the CF shall retrieve the previously stored D-END Response parameters and, if the D-END *Result* parameter has the abstract value "rejected":

a) Construct an A-RELEASE Response primitive with parameter values as follows:

Table 4.3-56.

A- RELEASE Response parameter	ISO Status	ATN Value
Reason	U	“not finished”
User Information	U	APDU received from the Security ASO
Result	M	“negative”

b) Invoke the A-RELEASE Response primitive; and

c) Remain in the RELEASE PENDING state.

4.3.3.8.1.2.9 When an APDU is validly submitted by the Security ASO, the CF is the Release Responder CF, is in the RELEASE PENDING state, and no D-END primitive was received, the CF shall:

a) Using the definition of presentation-user-data in 4.3.2.6, encode the APDU received from the Security ASO with presentation-context-identifier value corresponding to “user-ase-apdu”;

b) Invoke a P-DATA Request primitive with the resulting encoding as User Data; and

c) Remain in the same state.

4.3.3.8.1.2.10 When an APDU is validly submitted by the Security ASO and the CF is in the RELEASE COLLISION state, the CF shall:

a) Construct an A-RELEASE Response primitive with parameter values as follows:

Table 4.3-57.

A- RELEASE Response parameter	ISO Status	ATN Value
Reason	U	“normal”
User Information	U	APDU received from the Security ASO
Result	M	“affirmative”

b) Invoke the A-RELEASE Response primitive; and

c) Remain in the RELEASE COLLISION state.

4.4 SESSION LAYER REQUIREMENTS

Note.— The session layer requirements are described in many cases by means of completed protocol implementation conformance statement (PICS) proforma tables. In such tables, the "Ref." column contains a reference to the relevant section in the session layer PICS proforma, ISO/IEC 8327-2 | ITU-T Rec. X.245 (1995).

4.4.1 Protocol versions implemented

4.4.1.1 Session protocol versions shall be supported as specified in Table 4.4-1.

Table 4.4-1. Session Protocol Versions Supported

Ref.	**Version**	**ISO Status**	**ATN Support**
S.A.3/1	Version 1	O.1	-
S.A.3/2	Version 2	O.1	M

O.1: The ISO PICS requires that the implementation of one, and only one, version of the protocol is described.

4.4.2 Session Functional units

4.4.2.1 Session functional units (S-FUs) shall be selected as specified in Table 4.4-2.

Table 4.4-2. Selection of Session functional units

Ref.	**Functional unit**	**ISO Status**	**ATN Support**
S.A.6.1/1	Kernel	M	M
S.A.6.1/2	Negotiated release	O	X
S.A.6.1/3	Half Duplex (HD)	O.2	X
S.A.6.1/4	Duplex	O.2	M
S.A.6.1/5	Expedited Data (EX)	O	X
S.A.6.1/6	Typed Data	O	X
S.A.6.1/7	Capability Data Exchange	C1	X
S.A.6.1/8	Minor Synchronize (SY)	O	X
S.A.6.1/9	Symmetric Synchronize (SS)	O	X
S.A.6.1/10	Data Separation	C2	X
S.A.6.1/11	Major Synchronize	O	X
S.A.6.1/12	Resynchronise	O	X
S.A.6.1/13	Exceptions	C3	X
S.A.6.1/14	Activity Management (ACT)	O	X
See note	No-orderly release (NOR)	O	M
See note	Special User-data	O	X

Functional units added by efficiency enhancement amendment.

O.2: The ISO standard requires at least one of the functional units Duplex and Half Duplex to be implemented.

C1:if [S-FU(ACT)] then O else N/A

C2:if [S-FU(SY) or S-FU(SS)] then O else N/A

C3:if [S-FU(HD)] then O else N/A

4.4.3 Protocol mechanisms

4.4.3.1 Session protocol mechanisms shall be supported as specified in Table 4.4-3.

Table 4.4-3. Session Protocol Mechanisms Supported

Ref.	**Mechanism**	**ISO Status**	**ATN Support**	**Associated mnemonic**
S.A.6.2/1	Use of transport expedited data (Extended control Quality of Service)	C4	X	S-EXP_T
S.A.6.2/2	Reuse of transport connection	O	O	S-REUSE_T
S.A.6.2/3	Basic concatenation	M	M (Note 2)	
S.A.6.2/4	Extended concatenation (sending)	O	X	
S.A.6.2/5	Extended concatenation (receiving)	O	X	S-XCONC_RCV
S.A.6.2/6	Segmenting (sending)	O	X	S-SEG_SDR
S.A.6.2/7	Segmenting (receiving)	O	X	S-SEG_RCV
S.A.6.2/8	Max. size of SS-user-data (S-CONNECT) > 512	O	O	S-MAXSIZE_512
S.A.6.2/9	Max. size of SS-user-data (S-CONNECT) > 10240	O	O	S-MAXSIZE_10240
S.A.6.2/10	Max. size of SS-user-data (S-ABORT) >9	O	X	S-MAXSIZE_9
See note 1	Null-encoding protocol option	-	M	
See note 1	Short-connect protocol option	-	M	
See note 1	Short-encoding protocol option	-	X	

Note 1.— Protocol options added by efficiency enhancement amendment.

Note 2.— Only Category 1 SPDUs are used for this ATN profile. By definition, these are never concatenated. Therefore, Basic concatenation is not applicable to this specification, but is supported to the extent necessary for compliance with the ISO PICS.

C4:if [S-FU(EX)] then M else O

4.4.3.2 The session protocol shall implement the efficiency enhancements in ISO/IEC 8327-1:1996/Amd. 1:1997 | ITU-T Rec. X.225 (1995)/Amd.1 (1997) as specified, together with all approved amendments and defect report resolutions.

4.4.3.3 If the null encoding protocol option is offered by the initiating Session Protocol machine (SPM), the responding SPM shall select only the kernel, full-duplex and no-orderly release functional units for use on this connection.

4.4.3.4 Session Protocol Data Units (SPDUs) associated with the Short-connect protocol option (i.e. Short Connect (SCN), Short Accept (SAC), Short Accept Continue (SACC), Short Refuse (SRF) and Short Refuse Continue (SRFC)) shall be transferred as User-data on the Transport layer T-CONNECT primitives, where possible.

Note.— This is only possible if the complete SPDUs, including any User-data, meet any size restrictions of the T-CONNECT User-data.

4.4.4 Supported Roles

4.4.4.1 Session Connection

4.4.4.1.1 The roles for Session Connection shall be supported as specified in Table 4.4-4.

Table 4.4-4. Session Connection Roles Supported

Ref.	Role	ISO Status	ATN Support	Mnemonic
S.A.7.1.1.1/1	Connection initiator	O.3	M	S-CON_initiator
S.A.7.1.1.1/2	Connection responder	O.3	M	S-CON_responder

O.3: The ISO standard requires a conforming implementation to support at least one of these roles as required by the implementation.

4.4.4.1.2 When a connection establishment request is accepted, the SHORT-CPA PPDU in the SS-User-data of the positive S-CONNECT response/confirmation primitive shall map to the User-data parameter of a SAC SPDU.

4.4.4.1.3 When a connection establishment request is refused, the SHORT-CPR PPDU in the SS-User-data of the negative S-CONNECT response/confirmation primitive shall map to the User-data parameter of a SRF SPDU.

4.4.4.2 Orderly release

4.4.4.2.1 The roles for Session Orderly Release shall be supported as specified in Table 4.4-5.

Table 4.4-5. Session Orderly Release Roles Supported

Ref.	Role	ISO Status	ATN Support	Mnemonic
S.A.7.1.1.2/1	Requestor	O.4	N/A (See note)	S-REL_requestor
S.A.7.1.1.2/2	Acceptor	O.4	N/A (See note)	S-REL_acceptor

O.4: The ISO standard requires a conforming implementation to support at least one of these roles as part of the Kernel functional unit. However, selection of the No Orderly Release functional unit removes this requirement.

Note.— Not applicable, as the No Orderly Release functional unit is selected. For ATN applications, orderly release is provided by the CF as described in 4.3.

4.4.4.3 Normal Data Transfer

4.4.4.3.1 The roles for Session Normal Data Transfer shall be supported as specified in Table 4.4-6.

Table 4.4-6. Session Normal Data Transfer Roles Supported

Ref.	**Role**	**ISO Status**	**ATN Support**	**Mnemonic**
S.A.7.1.1.3/1	Requestor	O.5	M	S-DATA_requestor
S.A.7.1.1.3/2	Acceptor	O.5	M	S-DATA_acceptor

O.5: The ISO standard requires a conforming implementation to support at least one of these roles.

4.4.5 Supported SPDUs

Note.— This section specifies the SPDUs associated with the supported Session functional units. There are no additional SPDUs associated with the Duplex functional unit, or with the No Orderly Release functional unit.

4.4.5.1 Support for the SPDUs associated with the Kernel functional unit

4.4.5.1.1 Support for SPDUs shall be as specified in Table 4.4-7.

Table 4.4-7. Supported Session Protocol Data Units

		Sender		**Receiver**		
Ref.	**SPDU**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	**Mnemonics**
S.A.7.1.2/1	Connect (CN)	C5	N/A (Note 4)	C6	N/A (Note 4)	
S.A.7.1.2/2	Overflow Accept (OA)	C7	N/A (Note 4)	C8	N/A (Note 4)	S-OA_SDR / S-OA_RCV
S.A.7.1.2/3	Connect Data Overflow (CDO)	C9	N/A (Note 4)	C10	N/A (Note 4)	S-CDO_SDR / S-CDO_RCV
S.A.7.1.2/4	Accept (AC)	C6	N/A (Note 4)	C5	N/A (Note 4)	
S.A.7.1.2/5	Refuse (RF)	C6	N/A (Note 4)	C5	N/A (Note 4)	
S.A.7.1.2/6	Finish (FN)	C11	N/A (Note 2)	C12	N/A (Note 2)	
S.A.7.1.2/7	Disconnect (DN)	C12	N/A (Note 2)	C11	N/A (Note 2)	
S.A.7.1.2/8	Abort	M	N/A (Note 3)	M	N/A (Note 3)	
S.A.7.1.2/9	Abort Accept (AA)	O	N/A (Note 3)	M	N/A (Note 3)	
S.A.7.1.2/1 0	Data Transfer (DT)	C13	N/A (Note 3)	C14	N/A (Note 3)	
S.A.7.1.2/1 1	Prepare (PR)	C15	X	C15	X	S-PR_SDR / S-PR_RCV
See note 1	Short Connect (SCN)	C17	M	C17	M	
See note 1	Short Accept (SAC)	C17	M	C17	M	
See note 1	Short Refuse (SRF)	C17	M	C17	M	

		Sender		Receiver		
Ref.	**SPDU**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	**Mnemonics**
See note 1	Null (NL)	C18	M	C18	M	
See note 1	Short Connect Continue (SCNC)	C16	N/A	C16	N/A	
See note 1	Short Accept Continue (SACC)	C17	M	C17	M	
See note 1	Short Refuse Continue (SRFC)	C17	M	C17	M	
See note 1	Short Finish (SFN)	C16	N/A	C16	N/A	
See note 1	Short Disconnect (SDN)	C16	N/A	C16	N/A	
See note 1	Short Data Transfer (SDT)	C16	N/A	C16	N/A	
See note 1	Short Abort (SAB)	C16	N/A	C16	N/A	

Note 1.— PDUs defined in efficiency enhancement amendment.

Note 2.— Not applicable, as the no-orderly-release functional unit is selected.

Note 3.— Not applicable, as the null-encoding protocol option is selected.

Note 4.— Not applicable, as the short-connect protocol option is selected.

C5: if [S-CON_initiator] then M else N/A
C6: if [S-CON_responder] then M else N/A
C7: if [S-V1 or (NOT S-CON_responder)] then N/A else if [S-MAXSIZE_10240] then M else O
C8: if [NOT S-V1 and S-CON_responder and S-MAXSIZE_10240] then M else N/A
C9: if [S-V1 or (NOT S-CON_initiator)] then N/A else if [S-MAXSIZE_10240] then M else O
C10: if [NOT S-V1 and S-CON_initiator and S-MAXSIZE_10240] then M else N/A
C11: if [S-REL_requestor] then M else N/A
C12: if [S-REL_acceptor] then M else N/A
C13: if [S-DATA_requestor] then M else N/A
C14: if [S-DATA_acceptor] then M else N/A
C15: if [NOT S-V1 and S-MAXSIZE_9 and S-EXP_T] then M else N/A
C16: used only if the short-encoding protocol option is selected.

C17: used if short-encoding or null-encoding is used.

C18: used only if the null-encoding protocol option is supported.

4.4.5.1.2 SCN, SAC, SRF, SACC and SRFC SPDUs shall be encoded such that the parameter bit of the SI&P octet is set to the value 0, indicating that all following octets are User-information (i.e. no SPDU parameters are present).

Note.— This is a requirement of the null-encoding protocol option.

4.4.5.2 Support for the SPDUs associated with Token Exchange

4.4.5.2.1 Support for Session protocol data units associated with Token exchange shall be as specified in Table 4.4-8.

Table 4.4-8. SPDUs associated with Token Exchange

		Sender		**Receiver**	
Ref.	**SPDU**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
S.A.7.1.3/1	Give Tokens (GT)	M	- (See note 2)	M	- (See note 2)
S.A.7.1.3/2	Please Tokens (PT)	M	- (See note 2)	M	- (See note 2)

Note 1.— The ISO PICS states that these two SPDUs are used for Token Exchange, but they are also used as category 0 SPDUs in basic concatenation. Therefore, their implementation is mandatory even if no token is supported (reference ISO/IEC 8327-1 | ITU-T Rec. X.225 clauses 7.16 and 7.17). However, if the null-encoding protocol option is selected, their encoding will be null, i.e. not present.

Note 2.— Not applicable, as the null-encoding protocol option is selected.

4.4.6 Use of null-encoding and short-connect protocol options

4.4.6.1 The null-encoding and short-connect session protocol options shall be selected for use, with the requirements as specified in Table 4.4-9.

Table 4.4-9. Use of the null encoding and short-connect Session protocol options

Ref.	**Requirement**	**Base Status**	**ATN Requirement**
a	The calling and called session selectors are null	C1	M
b	The session-requirements parameter in the S-CONNECT service includes the kernel, full-duplex and no-orderly-release functional units only.	C1	M

C1: The SPMs may use the short-connect protocol option to establish a session connection using the null-encoding option. The null-encoding protocol option is available for use on an established connection only if the conditions a and b in Table 4.4-9 are both true.

4.4.7 Mapping to the ATN Internet Transport Service

4.4.7.1 The use of the connection-oriented transport service provided by the ATN Internet shall be as specified in Clause 6 of ISO/IEC 8327-1 | ITU-T Rec. X.225 (1995), except as stated in this section.

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

4.4.7.3 The TS-user shall indicate in all T-CONNECT requests that the transport expedited flow is not required.

4.4.7.4 Information on the use of the transport checksum shall be conveyed between the TS-User and TS-Provider via the "residual error rate" component of the T-CONNECT quality of service parameter.

Note 1.— 5.5.1.2 specifies the use by the TS-user of the required residual error rate parameter. This affects the degree of integrity checking of all data sent over the underlying transport connection.

Note 2.— In the ATN, the Quality of Service provided to applications is otherwise maintained using capacity planning techniques that are outside of the scope of this specification. Network administrators are responsible for designing and implementing a network that will meet the QOS requirements of the CNS/ATM applications that use it.

4.4.7.5 The use of the transport checksum shall be specified on a per Transport Connection basis, based on TS-User requests in the T-CONNECT request primitive.

4.4.7.6 The Application Service Priority shall be provided to the TS-Provider on a per Transport Connection basis, by implementation-specific means, and using the values for "Transport Layer Priority" specified in Table 1-2.

Note.— Although transport priority and network priority are semantically independent of each other, it is required (in 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 1-2, which defines the fixed relationship between transport priority and the network priority.

4.4.7.7 The ATN Security Label shall be provided to the TS-Provider on a per Transport Connection basis.

4.4.7.8 The ATN Security Label value shall be encoded according to 5.6.2.2.2.2 b), and passed between TS-User and TS-Provider by implementation-specific means.

4.4.7.9 The QOS parameter "Routing Class" shall be conveyed as the Security Tag field of the security tag set for Traffic Type and Associated Routing Policies within the ATN Security Label.

Note 1.— 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."

Note 2.— 5.5.1.2 requires the TS-User to provide the ATN Security Label as specified in Figure 5.6-1 and 5.6.2.2.2.2 b). The encoding of the ATN Security Label is summarised below. The D-START QOS parameter "Routing Class" maps to the field labelled "Traffic Type & category".

ATN Security Label field	***Value (Hex)***	***Length (Octets)***
Security Registration ID Length	*6*	*1*
Security Registration ID = OID {1.3.27.0.0}	*06, 04, 2B, 1B, 00, 00*	*6*
Security Information Length	*4*	*1*
Security information:		
Tag Set Name Length	*1*	*1*
Tag Set Name = "Traffic Type & Associated Routing Policies"	*0F*	*1*
Tag Set Length	*1*	*1*
Security Tag Value = Traffic Type & category (from Table 5.6-1)	*01 (for example)*	*1*
	Total:	*12 Octets*

4.4.7.10 No Transport Service quality of service parameters other than those specified in the preceding subsections shall be specified when establishing a transport connection.

4.5 PRESENTATION LAYER REQUIREMENTS

Note.— The presentation layer requirements are described in many cases by means of completed protocol implementation conformance statement (PICS) proforma tables. In such tables, the "Ref." column contains a reference to the relevant section in the presentation layer PICS proforma ISO/IEC 8823-2 | ITU-T Rec. X.246 (1996).

4.5.1 Protocol mechanisms

4.5.1.1 The Presentation protocol mechanisms supported shall be as specified in Table 4.5-1.

Table 4.5-1. Presentation Protocol Mechanisms Supported

Ref.	**Protocol Mechanism**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.6.1/2	Normal mode	O.1	M	
P.A.6.1/1	X.410-1984 mode	O.1	X	
See note	Nominated context	O	N/A	
See note	Short encoding	O	N/A	
See note	Packed encoding rules	O	N/A	
See note	Short-connect	O	M	
See note	Null-encoding	O	M	

Note.— Optional protocol mechanisms defined in efficiency enhancement amendment.

O.1: The ISO standard requires that either Normal mode or X.410 (1984) mode or both be supported.

4.5.1.2 The presentation protocol shall implement the efficiency enhancements in ISO/IEC 8823-1: 1994/Amd. 1: 1997 | ITU-T Rec. X.226 (1994)/Amd.1 (1997) as specified, together with all approved amendments and defect report resolutions.

4.5.2 Use of null-encoding and short-connect protocol options

4.5.2.1 The null-encoding and short-connect presentation protocol options shall be selected for use, with the requirements as specified in Table 4.5-2.

Table 4.5-2. Use of the null encoding and short-connect Presentation protocol options

Ref.	Requirement	Base Status	ATN Requirement
a	The presentation context definition list contains precisely one item in which the abstract syntax is known to the responding Presentation Protocol Machine (PPM) by bilateral agreement.	C1	N/A
b	The presentation context definition list is empty and the default context is known by bilateral agreement	C1	M
c	The presentation context definition list is empty and the abstract syntax of the default context is known to the responding PPM by bilateral agreement and is specified in ASN.1	C1	M
d	The calling and called presentation selectors are null	C2	M
e	The presentation-requirements parameter in the P-CONNECT service includes the kernel functional unit only.	C2	M

C1: The null-encoding protocol option is available for use on an established connection only if at least one of the conditions a, b and c in Table 4.5-2 is true.

C2: The short-connect protocol option is used only in connection establishment to establish a connection on which the null-encoding option will be used; it can only be used if both of the conditions d and e in Table 4.5-2 is true.

4.5.3 Mapping of Presentation Primitives to the Null Encoding option

Note.— When the null-encoding presentation protocol option is selected, no presentation protocol control information is present once the connection has been established. Thus, no presentation PDUs are supported. The presentation connection is only terminated by the termination of the supporting session and transport connections.

4.5.3.1 The user of the presentation service shall not issue any presentation primitives other than P-CONNECT request, P-CONNECT response, P-DATA request and P-U-ABORT request.

4.5.3.2 When it is required to release the presentation connection, the presentation service user shall issue a P-U-ABORT request.

4.5.3.3 Any user data in a P-U-ABORT request shall be ignored by the presentation service provider.

4.5.4 Functional units

4.5.4.1 The Presentation functional units selected shall be as specified in Table 4.5-3.

Table 4.5-3. Selection of Presentation functional units

Ref.	**Presentation functional unit**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.6.2/1	Kernel	M	M	
P.A.6.2/2	Presentation Context Management	O	X	P-FU(CM)
P.A.6.2/3	Presentation Context Restoration	C1	X	P-FU(CR)

C1: if Presentation Context Management (2) is supported then O else N/A

4.5.4.2 The Presentation pass-through functional units selected shall be as specified in Table 4.5-4.

Table 4.5-4. Selection of Presentation pass-through functional units

Ref.	**Pass-through to Session functional units**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.6.2/4	Negotiated release	O	X	S-FU(NR)
P.A.6.2/5	Half Duplex	O.2	X	S-FU(HD)
P.A.6.2/6	Duplex	O.2	M	S-FU(FD)
P.A.6.2/7	Expedited Data	O	X	S-FU(EX)
P.A.6.2/8	Typed Data	O	X	S-FU(TD)
P.A.6.2/9	Capability Data Exchange	C1	X	S-FU(CD)
P.A.6.2/10	Minor Synchronize	O	X	S-FU(SY)
P.A.6.2/11	Symmetric Synchronize	O	X	S-FU(SS)

Ref.	**Pass-through to Session functional units**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.6.2/12	Data Separation	O	X	S-FU(DS)
P.A.6.2/13	Major Synchronize	O	X	S-FU(MA)
P.A.6.2/14	Resynchronise	O	X	S-FU(RESYNC)
P.A.6.2/15	Exceptions	C2	X	S-FU(EXCEP)
P.A.6.2/16	Activity Management	O	X	S-FU(ACT)
See note	No-orderly release (NOR)	-	M	S-FU(NOR)

Note.— The NOR Session functional unit is defined in the ISO Session service efficiency enhancement amendment.

O.2: The ISO standard requires that pass-through for at least one of the Session functional units Duplex and Half Duplex be supported.

C1: if [S-FU(ACT) then O else N/A

C2: if [S-FU(HD) then O else N/A

4.5.5 Elements of procedure

4.5.5.1 Supported roles

4.5.5.1.1 Presentation Connection

4.5.5.1.1.1 The supported roles for establishing Presentation connections shall be as specified in Table 4.5-5.

Table 4.5-5. Presentation Connection roles

Ref.	**Role**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.7.1.1.1/1	Initiator	O.3	M	P-CON_initiator
P.A.7.1.1.1/2	Responder	O.3	M	P-CON_responder

O.3: The ISO standard requires a conforming implementation to support at least one of these roles.

4.5.5.1.1.2 When a connection establishment request is accepted, the AARE (accepted) in the User-data of the positive P-CONNECT response/confirmation primitive shall map to the User-data parameter of a SHORT-CPA PPDU.

4.5.5.1.1.3 When a connection establishment request is refused, the AARE (rejected) in the User-data of the negative P-CONNECT response/confirmation primitive shall map to the User-data parameter of a SHORT-CPR PPDU.

4.5.5.1.2 Orderly release

4.5.5.1.2.1 The supported roles for the orderly release of Presentation connections shall be as specified in Table 4.5-6.

Table 4.5-6. Presentation Connection orderly release roles

Ref.	**Role**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.7.1.1.3/1	Requestor	O	N/A	P-REL_requestor
P.A.7.1.1.3/2	Acceptor	O	N/A	P-REL_acceptor

4.5.5.1.3 Normal Data

4.5.5.1.3.1 The supported roles for Normal Data shall be as specified in Table 4.5-7.

Table 4.5-7. Presentation Normal Data roles

Ref.	**Role**	**ISO Status**	**ATN Support**	**Mnemonic**
P.A.7.1.1.2/1	Requestor	O	M	P-DATA_requestor
P.A.7.1.1.2/2	Acceptor	O	M	P-DATA_acceptor

4.5.6 Supported Presentation Protocol Data Units (PPDUs)

Note.— This section specifies the PPDUs associated with the supported Presentation functional units. There are no additional PPDUs or additional pass-through functionality associated with the supported Session functional units.

4.5.6.1 Supported PPDUs associated with the Kernel services

The Presentation Protocol Data Units supported shall be as specified in Table 4.5-8.

Table 4.5-8. Supported Presentation Protocol Data Units

		Sender		**Receiver**		
Ref.	**PPDU**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	**Mnemonics**
P.A.7.1.2/1	Connect presentation (CP)	C3	N/A (Note 2)	C4	N/A (Note 2)	
P.A.7.1.2/2	Connect presentation accept (CPA)	C4	N/A (Note 2)	C3	N/A (Note 2)	S-OA_SDR / S-OA_RCV
P.A.7.1.2/3	Connect presentation reject (CPR)	C4	N/A (Note 2)	C3	N/A (Note 2)	S-CDO_SDR / S-CDO_RCV
P.A.7.1.2/4	Abnormal release provider (ARP)	M	N/A (Note 2)	M	N/A (Note 2)	
P.A.7.1.2/5	Abnormal release user (ARU)	O	N/A (Note 2)	M	N/A (Note 2)	
P.A.7.1.2/6	Presentation Data (TD)	C5	N/A (Note 2)	C6	N/A (Note 2)	
Note 1	Short Connect (SHORT-CP)	O	M	O	M	
Note 1	Short Connect Accept (SHORT-CPA)	O	M	O	M	
Note 1	Short Connect Reject (SHORT-CPR)	O	M	O	M	

Note 1.— PDUs defined in efficiency enhancement amendment.

Note 2.— PPDUs not applicable, as the short-connect and null-encoding protocol options are selected.

C3: if [P-CON_initiator] then M else N/A
C4: if [P-CON_responder] then M else N/A
C5: if [P-DATA_requestor] then M else N/A
C6: if [P-DATA_acceptor] then M else N/A

4.5.6.2 Structure and encoding of PPDUs

4.5.6.2.1 The SHORT-CP, SHORT-CPA and SHORT-CPR PPDUs shall have the encoding-choice bit-field set to “unaligned PER”.

4.6 ACSE SPECIFICATION

Note.— The ACSE requirements are described in many cases by means of completed protocol implementation conformance statement (PICS) proforma tables. In such tables, the "Ref." column contains a reference to the relevant section in the ACSE PICS proforma ISO/IEC 8650-2 | ITU-T Rec. X.247 (1996).

4.6.1 Protocol details

4.6.1.1 The specification of the ACSE protocol supported shall be as defined in Table 4.6-1.

Table 4.6-1. Identification of ACSE Protocol Specification

Identification of Protocol Specification	ATN Support	Comments
ISO/IEC 8650-1:1995/Amd.1:1997 | ITU-T Rec. X.227 (1994)/Amd.1 (1996)	M	See note

Note.— This is the second edition of the ACSE protocol specification.

4.6.2 Protocol versions

4.6.2.1 The version of the ACSE protocol supported shall be as specified in Table 4.6-2.

Table 4.6-2. Identification of ACSE Protocol version

Ref.		**ISO Status**	**ATN Support**	**Mnemonic**
A.A.4.2/1	Version 1	O.1	M	A-V1
A.A.4.2/2	Version 2	O.1		

O.1: The ISO PICS requires support of the implementation of only one version of the protocol to be described.

4.6.3 Supported roles

4.6.3.1 Association establishment

4.6.3.1.1 The supported roles for Association Establishment shall be as specified in Table 4.6-3.

Table 4.6-3. ACSE Roles for Association Establishment

Ref.	Capability	ISO Status	ATN Support	Mnemonic
A.A.6.1/1	Association initiator	O.2	See text	A-CON_initiator
A.A.6.1/2	Association responder	O.2	See text	A-CON_responder

O.2: The ISO standard requires a conforming implementation to support at least one of the roles.

4.6.3.1.2 Either one or both of the ACSE roles "Association initiator" or "Association responder" shall be supported.

4.6.3.2 Normal Release procedure

4.6.3.2.1 The supported roles for the Normal Release procedure shall be as specified in Table 4.6-4.

Table 4.6-4. ACSE Roles for Normal Release

Ref.	Role	ISO Status	ATN Support	Mnemonic
A.A.6.2/1	Initiator	O	See text	A-REL_requestor
A.A.6.2/2	Responder	O	See text	A-REL_acceptor

4.6.3.2.2 Either one or both of the ACSE Normal Release roles "Initiator" or "Responder" shall be supported.

4.6.3.2.3 The ACSE Release Responder shall be allowed to give a negative response, despite the fact that the session Negotiated Release functional unit is not selected for the association.

Note.— The above provision waives the ISO/IEC 8649 | ITU-T Rec. X.217 (1995) requirement that the Responder may give a negative response only if session Negotiated Release is selected. This is possible because, for ATN, the ACSE release PDUs do not map directly to the Presentation release service; they are re-mapped by the CF to P-DATA.

4.6.3.3 Abnormal Release procedure

4.6.3.3.1 The supported roles for the Abnormal Release procedure shall be as specified in Table 4.6-5.

Table 4.6-5. ACSE Roles for Abnormal Release

Ref.	**Role**	**ISO Status**	**ATN Support**	**Mnemonic**
A.A.6.3/1	Initiator	M	M	
A.A.6.3/2	Responder	M	M	

4.6.4 Protocol mechanisms

4.6.4.1 The ACSE protocol mechanisms supported shall be as specified in Table 4.6-6.

Table 4.6-6. ACSE Protocol Mechanisms Supported

Ref.	**Protocol Mechanism**	**ISO Status**	**ATN Support**	**Mnemonic**
A.A.7/1	Normal mode	O.4	M	
A.A.7/2	X.410-1984 mode	O.4	X	
A.A.7/2	Rules for extensibility	M	M	
A.A.7/4	Supports operation of Session version 2	O	M	S-O-SESS-V2

O.4: The ISO standard requires either Normal mode or X.410-1984 mode or both to be supported.

4.6.4.2 Extensibility and Encoding

4.6.4.2.1 For the purposes of this specification, the abstract syntax module defined in clause 9 of the ACSE protocol specification shall be augmented with the ASN.1 extensibility notation, as specified in ISO/IEC 8650-1: 1996/Amd. 1: 1997 | ITU-T Rec. X.227 (1994)/Amd.1 (1996).

4.6.4.2.2 The system shall support that encoding which results from applying the ASN.1 packed encoding rules (basic, unaligned variant), as specified in ISO/IEC 8825-2 | ITU-T Rec. X.691 (1995), to the abstract syntax module specified in 4.6.4.2.1.

4.6.4.2.3 Packed encoding (basic, unaligned) shall be used for encoding all ACSE Protocol Control Information (PCI) for interchange.

Note.— When embedded in Fully-encoded-data at the Presentation Service boundary, encoded ACSE APDUs are treated as bit-oriented values that are not padded to an integral number of octets; the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

4.6.5 ACSE Functional units

4.6.5.1 The ACSE functional units selected shall be as specified in Table 4.6-7.

Table 4.6-7. Selection of ACSE Functional Units

Ref.	Role	ISO Status	ATN Support	Mnemonic
A.A.8/1	Normal mode	M	M	
A.A.8/2	Authentication	O	C1	A-FU(AU)

C1: If the Dialogue Service user requires the use of the Security Requirements parameter of the D-START primitives, then M, else O.

4.6.6 Supported APDUs

4.6.6.1 The ACSE Protocol data units supported shall be as specified in Table 4.6-8.

Table 4.6-8 Supported ACSE Protocol Data Units

Ref.	**APDU**	**Sender**		**Receiver**		**Comment**
		ISO Status	**ATN Support**	**ISO Status**	**ATN Support**	
A.A.9/ 1	AARQ	C1	M	C2	M	
A.A.9/ 2	AARE	C2	M	C1	M	
A.A.9/ 3	RLRQ	C3	M	C4	M	
A.A.9/ 4	RLRE	C4	M	C3	M	
A.A.9/ 5	ABRT	C5	M	C5	M	

C1: if [A-CON_initiator] then M else N/A
C2: if [A-CON_responder] then M else N/A
C3: if [A-REL_requestor] then M else N/A
C4: if [A-REL_acceptor] then M else N/A
C5: if [S-O-SESS-V2] then M else N/A

4.6.6.2 Supported APDU parameters

4.6.6.2.1 A-Associate-request (AARQ)

4.6.6.2.1.1 The parameters in the AARQ APDU shall be supported as specified in Table 4.6-9.

Table 4.6-9. Supported AARQ Parameters

		Sender		**Receiver**	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.10.1/1	Protocol Version	C6	X	C2	M
A.A.10.1/2	Application Context Name	C1	M	C2	M
A.A.10.1/3	Calling AP title	C6	M	C2	M
A.A.10.1/4	Calling AE qualifier	C6	M	C2	M
A.A.10.1/5	Calling AP invocation-identifier	C6	X	C2	M

		Sender		Receiver	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.10.1/6	Calling AE invocation-identifier	C6	X	C2	M
A.A.10.1/7	Called AP title	C6	X	C2	M
A.A.10.1/8	Called AE qualifier	C6	X	C2	M
A.A.10.1/9	Called AP invocation-identifier	C6	X	C2	See text
A.A.10.1/10	Called AE invocation-identifier	C6	X	C2	See text
A.A.10.1/11	ACSE-requirements	C8	See text	C9	M
A.A.10.1/12	Authentication-mechanism-name	C8	See text	C9	M
A.A.10.1/13	Authentication-value	C8	See text	C9	M
A.A.10.1/14	Implementation information	C6	X	C7	O
A.A.10.1/15	User information	C6	M	C7	M

C1: if [A-CON_initiator] then M else N/A
C2: if [A-CON_responder] then M else N/A
C6: if [A-CON_initiator] then O else N/A
C7: if [A-CON_responder] then O else N/A
C8: if [A-CON_initiator and A-FU(AU)] then M else N/A
C9: if [A-CON_responder and A-FU(AU)] then M else N/A

4.6.6.2.1.2 The AARQ parameters "ACSE-Requirements," "Authentication Mechanism-name" and "Authentication-value" shall be supported, for sending, only if the connection initiator role (A-CON_initiator) and the Authentication functional unit (A-FU(AU)) are supported.

4.6.6.2.1.3 The AARQ parameters "ACSE-Requirements," "Authentication Mechanism-name" and "Authentication-value" shall be supported for receiving if the connection responder role (A-CON_responder) is supported, but are ignored if the Authentication functional unit (A-FU(AU)) is not supported by the responder.

Note.— The ATN specification is non-conformant to the ISO PICS proforma, in that "ACSE-requirements," "Authentication Mechanism-name" and "Authentication-value" are "M" for receiving, even if the Authentication functional unit is not supported.

4.6.6.2.1.4 The AARQ parameters "Called AP invocation-identifier" and "Called AE invocation-identifier" shall be supported, for receiving, if the Association Responder role is supported.

4.6.6.2.2 A-Associate-response (AARE)

4.6.6.2.2.1 The parameters in the AARE APDU shall be supported as specified in Table 4.6-10.

Table 4.6-10. Supported AARE Parameters

		Sender		**Receiver**	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.10.2/1	Protocol Version	C7	X	C1	M
A.A.10.2/2	Application Context Name	C2	M	C1	M
A.A.10.2/3	Responding AP title	C7	X	C1	M
A.A.10.2/4	Responding AE qualifier	C7	X	C1	M
A.A.10.2/5	Responding AP invocation-identifier	C7	X	C1	M
A.A.10.2/6	Responding AE invocation-identifier	C7	X	C1	M
A.A.10.2/7	Result	C2	M	C1	M
A.A.10.2/8	Result source - diagnostic	C10	M	C11	M
A.A.10.2/9	ACSE-requirements	C9	See text	C8	See text
A.A.10.2/10	Authentication-mechanism-name	C9	See text	C8	See text
A.A.10.2/11	Authentication-value	C9	See text	C8	See text
A.A.10.2/12	Implementation information	C7	X	C6	O
A.A.10.2/13	User information	C7	M	C6	M

C1: if [A-CON_initiator] then M else N/A
C2: if [A-CON_responder] then M else N/A
C6: if [A-CON_initiator] then O else N/A
C7: if [A-CON_responder] then O else N/A
C8: if [A-CON_initiator and A-FU(AU)] then M else N/A
C9: if [A-CON_responder and A-FU(AU)] then M else N/A
C10: if [A-CON_responder] then (if [A-FU(AU)] then M (with a value range of 0 to 14) else M (with a value range of 0 to 10) else N/A
C11: if [A-CON_initiator] then (if [A-FU(AU)] then M (with a value range of 0 to 14) else M (with a value range of 0 to 10) else N/A

4.6.6.2.2.2 The AARE parameters “ACSE-Requirements”, “Authentication-mechanism-name” and “Authentication-value” shall be supported, for sending, only if the connection responder role (A-CON_responder) and the Authentication functional unit (A-FU(AU)) are supported.

4.6.6.2.2.3 The AARE parameters “ACSE-Requirements,” “Authentication-mechanism-name” and “Authentication-value” shall be supported, for receiving, only if the connection initiator role (A-CON_initiator) and the Authentication functional unit (A-FU(AU)) are supported.

4.6.6.2.3 A-Release-request (RLRQ)

4.6.6.2.3.1 The parameters in the RLRQ APDU shall be supported as specified in Table 4.6-11.

Table 4.6-11. Supported RLRQ Parameters

		Sender		**Receiver**	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.10.3/1	Reason	C12	M	C4	M
A.A.10.3/2	User information	C12	M	C4	M

C4: if [A-REL_acceptor] then M else N/A
C12: if [A-REL_requestor] then O else N/A

4.6.6.2.4 A-Release-response (RLRE)

4.6.6.2.4.1 The parameters in the RLRE APDU shall be supported as specified in Table 4.6-12.

Table 4.6-12 Supported RLRE Parameters

		Sender		**Receiver**	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.10.4/1	Reason	C13	M	C3	M
A.A.10.4/2	User information	C13	M	C3	M

C3: if [A-REL_requestor] then M else N/A
C13: if [A-REL_acceptor] then O else N/A

4.6.6.2.5 A-Abort (ABRT)

4.6.6.2.5.1 The parameters in the ABRT APDU shall be supported as specified in Table 4.6-13.

Table 4.6-13. Supported ABRT Parameters

		Sender		Receiver	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.10.5/1	Abort source	M	M	M	M
A.A.10.5/2	Diagnostic	C14	M	C14	M
A.A.10.5/3	User information	O	M	M	M

C14: if [A-FU(AU)] then M else N/A

4.6.6.3 Supported parameter forms and values

4.6.6.3.1 AE title name form

4.6.6.3.1.1 The Application Entity Title parameter shall be supported in the forms specified in Table 4.6-14.

Table 4.6-14. AE Title Name Form

		Sender		Receiver	
Ref.	**Syntax form**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.11.1/1	Form 1 (Directory name)	O.5	X	M	O
A.A.11.1/2	Form 2 (Object identifier and integer)	O.5	M	M	M

O.5: The ISO standard requires a conforming implementation to support at least one of the forms.

4.6.6.3.2 Authentication value form

4.6.6.3.2.1 The Authentication-value parameter shall be supported in the forms specified in Table 4.6-15.

Table 4.6-15. Authentication-value Form

Prerequisite: A-FU(AU)

		Sender		Receiver	
Ref.	**Authentication value form**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.11.2/1	GraphicString	O.6	X	C14	N/A
A.A.11.2/2	BIT STRING	O.6	M	C14	M

		Sender		Receiver	
Ref.	**Authentication value form**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
A.A.11.3/3	EXTERNAL	O.6	X	C14	N/A
A.A.11.4/4	Other	O.6	X	C14	N/A

O.6: The ISO standard requires a conforming implementation to support at least one of the forms.
C14: if [A-FU(AU)] then M else N/A

4.6.6.3.2.2 If the authentication functional unit is supported, the BIT STRING form of encoding the ACSE Authentication-value field shall be used, with a bit-oriented encoding such that no additional padding bits are appended to the encoded value.

Note.— The above provision means that data values encoded by SESE, when embedded in ACSE APDUs, are treated by the CF as normal BIT STRING values, not in general an integral number of octets. Padding to an octet boundary only applies to the outermost Fully-encoded -data value that is passed across the Presentation Service boundary.

4.6.6.3.3 User information form

4.6.6.3.3.1 User information reference

4.6.6.3.3.1.1 The User information parameter shall use the forms of reference specified in Table 4.6-16.

Table 4.6-16. User information reference

		Sender		Receiver	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
	direct-reference	O	X	M	N/A
	indirect-reference	O	O (See Note)	M	M
	data-value-descriptor	O	X	M	N/A

Note.— Indirect-reference contains a presentation-context-id value as specified in Table 4.3-3 when the single-ASN-1-type encoding form is used, and is absent otherwise.

4.6.6.3.3.2 User information encoding type

4.6.6.3.3.2.1 The User information parameter encoding choice shall be as specified in Table 4.6-17.

Table 4.6-17. User information encoding choice

		Sender		**Receiver**	
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**
	single-ASN1-type	O	O	M	M
	octet-aligned	O	X	M	N/A
	arbitrary	O	O	M	M

4.6.6.3.3.2.2 **Recommendation**.— *The arbitrary form of encoding ACSE user-information should be used.*

Note.— The above Recommendation, if followed, produces a canonical encoding for a given user PDU, is consistent with the Fully-encoded-data wrapper used by the CF at the Presentation Service boundary, and provides optimal bit-efficiency.

4.6.6.3.3.2.3 When the 'arbitrary' (BIT STRING) form of encoding is used, a bit-oriented encoding shall be applied, such that no additional padding bits are appended to the encoded BIT STRING value in the EXTERNAL user information type, nor to the encoded EXTERNAL value itself.

Note.— The above provision means that data encoded by ATN-App ASEs, when embedded as user-information in ACSE APDUs, are treated by the CF as normal BIT STRING values, not in general an integral number of octets. Padding to an octet boundary only applies to the outermost Fully-encoded-data value that is passed across the Presentation Service boundary.

4.6.6.3.3.2.4 If the 'single-ASN1-type' (ABSTRACT-SYNTAX.&Type) form of encoding is used, the octet-oriented encoding of an open type shall be applied, such that additional padding bits are appended to make the length of the encoding produced so far a multiple of eight bits.

Note.— Encoding as single-ASN1-type is permitted for backward compatibility, but its use is deprecated.

4.6.6.3.4 Supported parameter values

4.6.6.3.4.1 The mechanism-name field in AARQ and AARE apdus shall be omitted when sending and ignored when receiving.

Note.— Use of the Authentication mechanism-name field is reserved for use in future versions of the ATN profile.

4.6.7 Mapping to the Presentation Service

4.6.7.1 The mapping of ACSE APDUs and parameters to presentation service primitives shall be performed by the CF as specified in 4.3, which takes precedence over the direct mapping defined in clause 8 of ISO/IEC 8650-1 | ITU-T Rec. X.227 (1994).

4.7 CONNECTIONLESS DIALOGUE SERVICE AND PROFILE

4.7.1 Scope of Connectionless Dialogue Service

Note 1.— The connectionless dialogue service allows the exchange of datagrams between communicating users. The connectionless dialogue service and supporting upper layer profiles are specified in this section.

Note 2.— The Session portion of the specified profile is based on the efficiency enhancements to the connectionless Session protocol which are standardised in ISO/IEC 9548-1 Amendment 1 | ITU-T Rec. X.235 (1995)/Amd.1 (1998).

Note 3.— The Presentation portion of the specified profile is based on the efficiency enhancements to the Presentation protocol which are standardised in ISO/IEC 9576-1 Amendment 1 | ITU-T Rec. X.236 (1995)/Amd.1 (1998).

Note 4.— As a consequence of using the Session and Presentation protocol efficiency enhancements, the protocol control information transferred by these protocols amounts to two octets per datagram.

Note 5.— The ACSE portion of the specified profile is based on ISO/IEC 10035-1 | ITU-T Rec. X.237 (1994), including the extensibility notation and authentication parameters as specified as Amendment 1 to that standard.

4.7.1.1 When it is required to use a connectionless mode supporting protocol stack, implementations of the ATN-App ASE, together with the UL elements which provide the Connectionless Dialogue Service (CLDS), shall exhibit the behaviour defined in this abstract service definition.

Note.— When using a connectionless mode supporting protocol stack, the ATN-App ASE will not benefit from the reliability and message sequencing characteristics offered by the connection oriented stack.

4.7.2 Service Primitives

4.7.2.1 Implementations which claim to support the CLDS functionality shall exhibit the behaviour defined by the service primitives in Table 4.7-1.

Table 4.7-1. Summary of Connectionless Dialogue Service primitives

Service	**Description**
D-UNIT-DATA	This unconfirmed service is used by a DS-User to send a datagram from that DS-User to a peer DS-User

Note.— Table 4.7-2 lists the parameters used when invoking the CLDS.

Table 4.7-2. Parameters of the Connectionless Dialogue Service primitives

Service	***Parameters***
D-UNIT-DATA	*Called Peer ID*
	Called Sys-ID
	Called Presentation Address
	Calling Peer ID
	Calling Sys-ID
	Calling Presentation Address
	DS-User Version Number
	Security Requirements
	Quality-of-Service
	User Data

4.7.2.2 Sequence of Primitives

4.7.2.2.1 Implementations which claim to support the CLDS functionality shall exhibit behaviour allowing two communicating DS-Users to send an item of User Data (a Datagram) from one DS-User to the other, consistent with the appropriate use of the corresponding service primitives.

4.7.3 The D-UNIT-DATA service

4.7.3.1 The behaviour defined by the D-UNIT-DATA service primitive shall be provided to enable the sending of a datagram from one DS-User to a peer DS-User.

Note 1.— D-UNIT-DATA is an unconfirmed service which is invoked by a DS-User to send data to a peer DS-User. D-UNIT-DATA request and indication primitives are defined, as illustrated in Figure 4.7-1

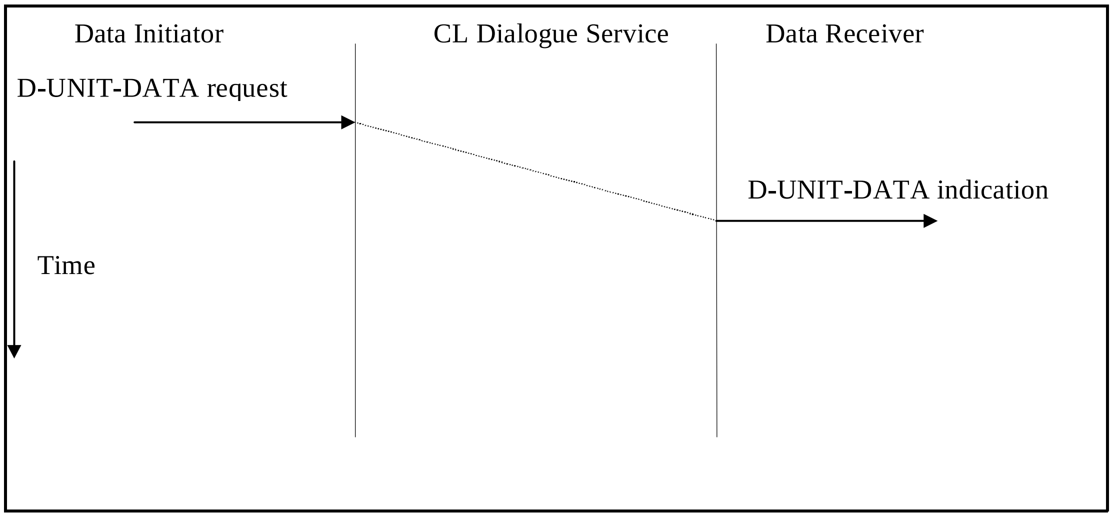


Figure 4.7-1. D-UNIT-DATA sequence diagram

Note 2.— The initiating DS-User issues a D-UNIT-DATA request primitive. The parameters of the D-UNIT-DATA primitives are specified in Table 4.7-3.

Table 4.7-3 D-UNIT-DATA parameters

Parameter Name	***Req***	***Ind***
Called Peer ID	*U*	
Called Sys-ID	*C*	
Called Presentation Address	*U*	
Calling Peer ID	*U*	*C(=)*
Calling Sys-ID	*C*	*C(=)*
Calling Presentation Address	*U*	*C(=)*
DS-User Version Number	*U*	*C(=)*

Parameter Name	***Req***	***Ind***
Security Requirements	*U*	*C(<=)*
Quality Of Service	*M*	*M(=)*
User Data	*M*	*M(=)*

Note 3.— Called Peer identification. The DS-User identifies the intended peer DS-User by specifying either a name or an address in the D-UNIT-DATA request primitive. The DS-User therefore specifies a value for one, and only one, of the Called Peer ID and Called Presentation Address parameters in the D-UNIT-DATA request. The Called Peer ID parameter is optionally used in the D-UNIT-DATA service to specify the name of the intended peer DS-User, and takes an abstract value corresponding to either a 24-bit ICAO aircraft-id or an ICAO facility designator. The Called Presentation Address parameter is optionally used in the D-UNIT-DATA service to specify the address of the intended peer DS-User, and takes the value of an ATN PSAP address (equivalent to an ATN TSAP address as defined in 5.4.2, since Session and Presentation selectors are NULL). If the Called Peer ID parameter is used, then the DS-User may additionally specify the Called Sys-ID parameter to explicitly refer to a specific instance of the peer DS-User at the remote location. If the Called Sys-ID parameter is absent, then ANY instance of the DS-User process at the identified location is being addressed. The syntax of the Called Sys-ID is an 8-octet identifier corresponding to the LOC + SYS fields of the ATN TSAP address, as defined in 5.4.3.8.

Note 4.— Calling Peer identification. The DS-User may optionally request that the name or address of the initiating DS-User be conveyed to the peer DS-User in the D-UNIT-DATA service. The DS-User therefore specifies a value for neither or one of the Calling Peer ID and Calling Presentation Address parameters in the D-UNIT-DATA request. If the Calling Peer ID parameter is used, then the DS-User may additionally specify the Calling Sys-ID parameter to explicitly refer to a specific instance of the local DS-User. If the Calling Sys-ID parameter is absent, then ANY instance of the DS-User process at the local location is being referenced. The presence of each of Calling Peer ID and Calling Sys-ID parameters in the D-UNIT-DATA indication primitive is conditional upon them being specified by the DS-User in the D-UNIT-DATA request primitive. If the Calling Peer ID is not specified in the D-UNIT-DATA request, then the Calling Presentation Address will be present in the D-UNIT-DATA indication, regardless of whether it was specified in the D-UNIT-DATA request. The syntax of the Calling Peer ID, Calling Sys-ID and Calling Presentation Address parameters is identical to the corresponding Called parameters described above.

Note 5.— The DS-User Version Number allows a DS-User to provide the peer with version information. The parameter is optional in the request primitive. Its presence in the indication primitive is conditional upon it being specified by the DS-User in the request primitive. If present, it may take any abstract value in the range 1 to 255.

Note 6.— The Security Requirements parameter allows a DS-User to provide the peer with security information. The parameter is optional in the request primitive. Its presence in the indication primitive is conditional upon it being specified by the DS-User in the request primitive .

Note 7.— The Quality Of Service parameter allows the initiating DS-User to specify in the request primitive its requirements for the quality of service (QOS) to be provided for the dialogue. For ATN, the parameter is not modified by the DS-provider, so the value in the indication primitive is equal to the value in the request. The following QOS parameters may be specified:

a) Routing Class - valid values are defined in Table 5.6-1.

b) Priority - valid values are defined in Table 1-2.

c) Residual Error Rate - valid values are defined in 5.5.3.3.4.

Note 8.— If the Routing Class parameter is not provided by the DS-User in the D-UNIT-DATA Request primitive, and the DS-User is an ATS application as specified in 2.1-2.4, then the default value "ATSC: No Traffic Type Policy Preference" is assumed. If the DS-User is not an ATS application as specified in 2.1-2.4, then the default traffic type "General Communications" is assumed.

Note 9.— If a Priority value is not provided by the DS-User in the D-UNIT-DATA Request primitive, then the default value "network/systems administration" is assumed.

Note 10.— For the RER parameter, a low error rate corresponds to a high degree of integrity checking, and a higher error rate corresponds to a lower degree of integrity checking. For ATSC applications, the highest available integrity level is always selected. Other types of application may select the required integrity level in the D-UNIT-DATA request, e.g. for compatibility with basic ISO | ITU-T Transport Service providers which do not support the ATN enhanced transport checksum.

Note 11.— The User Data parameter contains the data to be sent from one DS-User to a peer DS-User during the D-UNIT-DATA service invocation. The Transport CL user data is limited to 63488 octets per TSDU, so the D-UNIT-DATA User Data length is limited to 63488 minus one octet for each of session and presentation protocols, minus the CL ACSE and CF protocol overheads.

4.7.4 Control Function for the Connectionless Mode Dialogue Service

4.7.4.1 ATN-App CF State Definitions

4.7.4.1.1 The ATN-App AE shall behave as if it has a Control Function which can exist only in the NULL state (STA 0).

4.7.4.1.2 When using a connectionless supporting protocol stack, the ATN-App AE CF shall behave as if it has a control function in accordance with the state table specified in Table 4.7-4, which shows diagrammatically the state transitions and actions performed by the CF in response to incoming events.

Note.— The conventions used in Table 4.7-4 are as described in 4.3.3.1.2 for the connection oriented CF.

Table 4.7-4. ATN-App Connectionless CF State Table

Event Source	**State--> Event**	**STA 0 NULL**
From ATN-App ASE (lower)	D-UNIT-DATA req	STA 0 A-UNIT-DATA req
From ACSE (upper)	A-UNIT-DATA ind	STA 0 D-UNIT-DATA ind
From ACSE (lower)	P-UNIT-DATA req	STA 0 P-UNIT-DATA req
From supporting service	P-UNIT-DATA ind	STA 0 P-UNIT-DATA ind

4.7.4.1.3 Incoming events handled by the connectionless CF shall be as enumerated in Table 4.7-5.

Table 4.7-5. Incoming Event List

Abbreviated name	**Source**	**Description**
D-UNIT-DATA req	ATN-App ASE (lower service boundary)	D-UNIT-DATA Request primitive issued by DS-User
A-UNIT-DATA ind	ACSE (upper service boundary)	A-UNIT-DATA Indication primitive issued by ACSE service
P-UNIT-DATA req	ACSE (lower service boundary)	P-UNIT-DATA Request primitive issued by ACSE Protocol Machine (ACPM)
P-UNIT-DATA ind	Supporting service	P-UNIT-DATA Indication primitive issued by presentation service provider

4.7.4.1.4 The actions performed by the connectionless CF shall be as enumerated in Table 4.7-6.

Table 4.7-6. Outgoing Event List

Abbreviated name	**Target**	**Description**
D-UNIT-DATA ind	DS-User	D-UNIT-DATA Indication primitive issued.
A-UNIT-DATA req	ACSE service provider	A-UNIT-DATA Request primitive issued.
P-UNIT-DATA ind	Lower ACSE service boundary	P-UNIT-DATA Indication primitive issued.
P-UNIT-DATA-req	Supporting service	P-UNIT-DATA Request primitive issued.

4.7.4.2 Services Invoked by ATN-App ASE

4.7.4.2.1 D-UNIT-DATA Request primitive

4.7.4.2.1.1 When Invoked

4.7.4.2.1.1.1 When the D-UNIT-DATA Request primitive is invoked by the ATN-App ASE, a new instance of communication shall be created, with its CF initially in the NULL state

4.7.4.2.1.2 Action Upon Invocation

4.7.4.2.1.2.1 When the D-UNIT-DATA Request is validly invoked, the CF shall :

a) Determine the app-type as defined for the ATN-App AE.

b) Construct the Application Context Name, with the value of the “version” arc set equal to the DS-User Version Number parameter if provided, and set to zero otherwise.

c) If not specified in the request primitive, retrieve the local calling Presentation address.

d) Determine the Called Presentation Address either directly from the Called Presentation Address parameter if present, or via look-up from the Called Peer ID and Called Sys-ID parameters.

e) If the Calling Peer ID parameter is present, then retrieve the corresponding Calling AP Title. If, in addition to Calling Peer ID, the optional Calling Sys-ID parameter is present, then retrieve the corresponding Calling AE-Qualifier. If Calling Peer ID is not present, then calling AP Title and Calling AE-Qualifier are not used in the A-UNIT-DATA request (and they will not then be included in the resulting A-UNIT-DATA (AUDT) APDU).

Note.— The way that the calling AP Title and the Calling AE qualifier are retrieved is a local implementation matter.

f) If the Security Requirements parameter is not present, make no use of the A-UNIT-DATA parameter "ACSE Requirements". If the Security Requirements parameter is present, add the symbolic value "authentication" to the ACSE Requirements parameter; and map the Security Requirements value to the A-UNIT-DATA Authentication-value parameter.

g) Construct an A-UNIT-DATA Request primitive with the following parameters:

Table 4.7-7. A-UNIT-DATA request parameters

A-UNIT-DATA Request parameter	**ISO Status**	**ATN value**
Application Context Name	M	As derived in b) above
Calling AP Title	U	As derived in e) above
Calling AE Qualifier	U	As derived in e) above
Calling AP Invocation-identifier	U	Not used
Calling AE Invocation-identifier	U	Not used
Called AP Title	U	Not used
Called AE Qualifier	U	Not used
Called AP Invocation-identifier	U	Not used
Called AE Invocation-identifier	U	Not used
User Information	M	D-UNIT-DATA User Data parameter
Calling Presentation Address	M	Derived as in c) above
Called Presentation Address	M	Derived as in d) above
Presentation Context Definition List	U	Not used
Quality of Service	M	Derived as in following subsection
Authentication-mechanism-name	U	Not used
Authentication-value	U	As derived in f) above
ACSE Requirements	U	As derived in f) above

h) Invoke the A-UNIT-DATA Request primitive.

4.7.4.2.1.3 Quality of Service parameter mappings

Note.— The following paragraphs specify how the Quality of Service parameters in the D-UNIT-DATA Request primitive are conveyed to the ATN Internet.

4.7.4.2.1.3.1 The Routing Class component of the quality of service parameter in the D-UNIT-DATA Request primitive shall be conveyed to the ATN Internet and mapped to ATN Security Label by local means, using the values for Security Tag Value specified in Table 5.6-1.

Note.— 5.2.7.3.2 states that the mechanism by which ATN Security Labels are associated with Transport Service Data Units (TSDUs) in connectionless mode 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 Transport Service Access Point (TSAP), or be identified using a Systems Management function.

4.7.4.2.1.3.2 If no value for Routing Class is specified in the D-UNIT-DATA Request primitive, then a default value shall be assigned as follows:

a) If the ATN-App AE is one of the ATS applications specified in 2.1 - 2.4, the value corresponding to "ATSC: No Traffic Type Policy Preference" is assigned;

b) otherwise, the traffic type defaults to General Communications, and no Security Tag Value is conveyed.

4.7.4.2.1.3.3 The Priority component of the quality of service parameter in D-UNIT-DATA Request primitive shall be provided to the TS-Provider, by implementation-specific means, using the values for "Transport Layer Priority" specified in Table 1-2.

Note.— Although transport priority is not supported by the connectionless transport protocol, 5.5.3.3.2.1 allows the Transport Service (TS)-user to specify a priority value, which in turn is mapped into the resulting Connectionless Network Protocol (CLNP) PDUs according to Table 1-2, which defines the fixed relationship between transport priority and the network priority.

4.7.4.2.1.3.4 If no value for Priority is specified in the D-UNIT-DATA Request primitive, then the value corresponding to "Network/systems administration" shall be used.

4.7.4.2.1.3.5 If the Routing Class parameter has an ATSC value, then the residual error rate (RER) component of the A-UNIT-DATA Quality of Service parameter shall be set to the logical value that maps to the lowest RER (highest integrity) supported by the transport service.

Note.— The A-UNIT-DATA residual error rate is mapped to the T-UNIT-DATA RER parameter, the use of which is specified in 5.5.3.3.4.

4.7.4.2.1.3.6 If the Routing Class parameter has a non-ATSC value, then the residual error rate (RER) component of the quality of service parameter in D-UNIT-DATA Request primitives shall map to the residual error rate component of the A-UNIT-DATA Quality of Service parameter.

4.7.4.3 ACSE Services delivered to the CF

4.7.4.3.1 A-UNIT-DATA Indication primitive

4.7.4.3.1.1 When Invoked

4.7.4.3.1.1.1 The A-UNIT-DATA Indication primitive may be validly invoked by the ACSE Protocol Machine (ACPM) when the CF is in the NULL state; if it is in any other state then appropriate error recovery action shall be taken.

4.7.4.3.1.2 Action Upon Invocation

4.7.4.3.1.2.1 When an A-UNIT-DATA Indication primitive is validly invoked, the CF shall:

a) If the “version” component of the Application Context Name parameter is non-zero, then use it as the DS-User Version Number in the D-UNIT-DATA Indication primitive. If it has the value zero, then omit the DS-User Version Number parameter in the D-UNIT-DATA Indication.

b) If the Calling AP Title parameter is present, extract the Calling Peer Id from it. If the Calling AP Title contains an <app-type> arc, then extract the Calling Sys-ID from the Calling AE Qualifier, if present. If the Calling AP Title parameter is absent, extract the Calling Presentation Address from the A-UNIT-DATA Indication.

c) If the ACSE Requirements parameter is present, and it indicates that the authentication functional unit is requested, then extract the Authentication-value parameter.

d) Extract the User Data from the A-UNIT-DATA User Information parameter.

e) Construct a D-UNIT-DATA Indication primitive, with the following parameter values:

Table 4.7-8. D-UNIT-DATA indication parameters

D-UNIT-DATA Indication parameter	**Value**
Calling Peer Id	Derived as in b) above
Calling Sys-ID	Derived as in b) above

D-UNIT-DATA Indication parameter	**Value**
Calling Presentation Address	Derived as in b) above
DS-User Version Number	Derived as in a) above
Security Requirements	Derived as in c) above
Quality of Service	See following subsection
User Data	Derived as in d) above

f) Invoke the D-UNIT-DATA Indication primitive.

4.7.4.3.1.3 Quality of Service parameter mappings

Note.— The following paragraphs specify how the Quality of Service parameters in A-UNIT-DATA Indication primitives are conveyed to the DS-User as parameters of the D-UNIT-DATA Indication primitives.

4.7.4.3.1.3.1 The Routing Class component of the quality of service parameter in D-UNIT-DATA indication primitives shall be obtained from the ATN Internet by local means, using the abstract values for Security Tag Values as specified in Table 5.6-1.

4.7.4.3.1.3.2 The Priority component of the quality of service parameter in D-UNIT-DATA indication primitives shall be taken from information provided by the TS-Provider, by implementation-specific means, using the abstract values for "Transport Layer Priority" specified in Table 1-2.

4.7.4.3.1.3.3 The RER component of the quality of service parameter in D-UNIT-DATA indication primitives shall be taken from the residual error rate component of the A-UNIT-DATA Quality of Service parameter.

4.7.4.4 Services used by ACSE

4.7.4.4.1 P-UNIT-DATA Request primitive

4.7.4.4.1.1 When Invoked

4.7.4.4.1.1.1 The P-UNIT-DATA Request primitive may be validly invoked by the ACPM when the CF is in the NULL state; if it is in any other state then appropriate error recovery action shall be taken.

4.7.4.4.1.2 Action Upon Invocation

4.7.4.4.1.2.1 When a P-UNIT-DATA Request primitive is validly invoked, and the CF is in the NULL state, the CF shall transparently invoke the equivalent primitive at the presentation service boundary and remain in the same state.

4.7.4.5 Supporting Services Delivered to the CF

4.7.4.5.1 P-UNIT-DATA Indication primitive

4.7.4.5.1.1 When Invoked

4.7.4.5.1.1.1 When the P-UNIT-DATA Indication primitive is invoked by the supporting service, a new instance of communication shall be created, with its CF initially in the NULL state.

4.7.4.5.1.2 Action Upon Invocation

4.7.4.5.1.2.1 When a P-UNIT-DATA Indication primitive is validly invoked, the CF shall transparently invoke the equivalent presentation service primitive at the lower ACSE service boundary.

4.7.5 Subsetting Rules

4.7.5.1 Either one of both of the connection-oriented or the connectionless mode upper layer profiles, with the corresponding CFs specified in 4.3.3 and 4.7.4 respectively shall be implemented.

Note.— The ATS applications specified in 2.1 - 2.4 require implementation of the connection-oriented upper layer profile and CF.

4.7.5.2 **Recommendation.—** *Implementations which support both connection-oriented and connectionless profiles should be capable of being configured such that both profiles will either operate over the same TSAP, or over different TSAPs.*

4.7.6 APRL for Connectionless Session Protocol

4.7.6.1 Support for SPDUs associated with the connectionless mode shall be as specified in Table 4.7-9.

Table 4.7-9. Supported Connectionless Session Protocol Data Units

		Sender		**Receiver**		
Ref.	**SPDU**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	**Mnemonics**
1	Unit Data (UD)	O.1	N/A (Note 2)	O.1	N/A (Note 2)	
See note 1	Short Unit Data (SUD)	C1	M	C1	M	

O.1: The ISO PICS requires that a conforming implementation support at least one of the roles.

Note 1.— PDU defined in efficiency enhancement amendment.

Note 2.— Not applicable, as only the SHORT UNIT DATA choice of the SPM is selected for ATN use.

4.7.6.2 Supported parameters

4.7.6.2.1 SUD SPDU sender

4.7.6.2.1.1 Parameters of the SUD SPDU shall be supported for sending as specified in Table 4.7-10.

Table 4.7-10. SUD Parameters when Sending

Prerequisite: SCNLS-SUD-Sdr

	Parameter	**ISO Status**	**ATN Support**	**Mnemonic**
1	User information field	M	M	

4.7.6.2.2 SUD SPDU receiver

4.7.6.2.2.1 Parameters of the SUD SPDU shall be supported for receiving as specified in Table 4.7-11.

Table 4.7-11. SUD Parameters when Receiving

Prerequisite: SCNLS-SUD-Rcv

	Parameter	ISO Status	ATN Support	Mnemonic
1	User information field	M	M	

4.7.6.3 Mapping to the ATN Internet Connectionless Transport Service

4.7.6.3.1 The use of the connectionless mode transport service provided by the ATN Internet, and specified in 5.5.3, shall be as specified in Clause 5.2 of ISO/IEC 9548-1| ITU-T Rec. X.235 (1995), except as stated in this section.

4.7.6.3.2 The source and destination Transport Addresses shall be provided to the TS-Provider on a per T-UNIT-DATA basis, using the called and calling Presentation Addresses as provided to ACSE in the A-UNIT-DATA request, with null presentation and session selectors.

4.7.6.3.3 Information on the use of the transport checksum shall be conveyed between the TS-User and TS-Provider via the "residual error rate" component of the T-UNIT-DATA quality of service parameter.

4.7.6.3.4 The TSDU priority shall be provided to the TS-Provider, by implementation-specific means, using the values for "Transport Layer Priority" specified in Table 1-2.

Note.— Although transport priority is not supported by the connectionless transport protocol, 5.5.3.3.2.1 allows the Transport Service (TS)-user to specify a priority value, which in turn is mapped into the resulting Connectionless Network Protocol (CLNP) PDUs according to Table 1-2, which defines the fixed relationship between transport priority and the network priority.

4.7.6.3.5 The ATN Security Label shall be provided to the TS-Provider on a per T-UNIT-DATA basis.

4.7.6.3.6 The required ATN Security Label shall be conveyed by local means, using the encoding specified in 5.6.2.2.2.2.b).

4.7.6.3.7 The QOS parameter "Routing Class" shall be conveyed as the Security Tag field of the security tag set for Traffic Type and Associated Routing Policies within the ATN Security Label.

4.7.6.3.8 No Transport Service quality of service parameters other than those specified in the preceding subsections shall be specified when invoking the T-UNIT-DATA.

4.7.7 APRL for Connectionless Presentation Protocol

4.7.7.1 The connectionless Presentation protocol mechanisms supported shall be as specified in Table 4.7-12.

Table 4.7-12. Supported Connectionless Presentation Protocol Mechanisms

Ref.	**Protocol Mechanism**	**ISO Status**	**ATN Support**	**Mnemonics**
See note	Short encoding	O	M	

Note.— Optional protocol mechanism defined in efficiency enhancement amendment.

4.7.7.2 Support for PPDUs associated with the connectionless mode shall be as specified in Table 4.7-13.

Table 4.7-13. Supported Connectionless Presentation Protocol Data Units

		Sender		**Receiver**		
Ref.	**PPDU**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	**Mnemonics**
1	Unit Data (UD)	O.1	N/A (Note 2)	O.1	N/A (Note 2)	
See note 1	Short Unit Data (SUD)	C1	M	C1	M	

O.1: The ISO PICS requires that a conforming implementation support at least one of the roles.

Note 1.— PDU defined in efficiency enhancement amendment.

Note 2.— Not applicable, as the short encoding CL protocol option is selected for ATN use.

4.7.7.3 Supported parameters

4.7.7.3.1 SUD PPDU sender

4.7.7.3.1.1 Parameters of the SUD PPDU shall be supported for sending as specified in Table 4.7-14.

Table 4.7-14. SUD Parameters when Sending

Prerequisite: PCNLS-SUD-Sdr

	Presentation PDU parameter	ISO Status	ATN Support	Mnemonic
1	Encoding-choice	M	M	
2	User data	O	M	

4.7.7.3.2 SUD PPDU receiver

4.7.7.3.2.1 Parameters of the SUD PPDU shall be supported for receiving as specified in Table 4.7-15.

Table 4.7-15. SUD Parameters when Receiving

Prerequisite: PCNLS-SUD-Rcv

	Presentation PDU parameter	ISO Status	ATN Support	Mnemonic
1	Encoding-choice	M	M	
2	User data	O	M	

4.7.7.4 Structure and encoding of PPDUs

4.7.7.4.1 The SUD PPDU shall have the value of the encoding-choice bit-field set to "unaligned PER".

4.7.8 APRL for Connectionless ACSE Protocol

4.7.8.1 Extensibility and Encoding

4.7.8.1.1 For the purposes of this specification, the abstract syntax module defined in clause 9 of the connectionless ACSE protocol specification shall be augmented with the ASN.1 extensibility notation, as specified in ISO/IEC 10035-1 Amendment 1 | ITU-T Rec. X.237 (1994)/Amd.1 (1996).

4.7.8.1.2 The system shall support that encoding which results from applying the ASN.1 packed encoding rules (basic, unaligned variant), as specified in ISO/IEC 8825-2 | ITU-T Rec. X.691 (1995), to the abstract syntax module specified in 4.7.8.1.1.

4.7.8.1.3 Packed encoding (basic, unaligned) shall be used for encoding all connectionless ACSE PCI for interchange.

4.7.8.2 ACSE Functional units

4.7.8.2.1 The ACSE functional units selected shall be as specified in Table 4.7-16.

Table 4.7-16. Selection of ACSE Functional Units

Ref.	**Role**	**ISO Status**	**ATN Support**	**Mnemonic**
1	Kernel	M	M	
2	Authentication	O	C1	A-FU(AU)

C1: If the Connectionless Dialogue Service user requires the use of the Security Requirements parameter of the D-UNIT-DATA primitives, then M, else O.

4.7.8.3 Support for A-UNIT-DATA (AUDT) APDU

Note.— This clause is used to declare if the system is capable of initiating an AUDT APDU or reacting to an AUDT APDU or both. No association connection exists and there is no response to an AUDT APDU.

4.7.8.3.1 Support for ACSE APDUs associated with the connectionless mode shall be as specified in Table 4.7-17.

Table 4.7-17. Supported Roles for AUDT Application Protocol Data Unit

		ISO Status	ATN Support	Mnemonic
1	Sender	O.1	M	ACNLS-UD-Sdr
2	Receiver	O.1	M	ACLNS-UD-Rcv

O.1: the ISO PICS states that a conforming implementation supports at least one of the roles

4.7.8.4 Supported AUDT parameters

4.7.8.4.1 AUDT APDU sender

4.7.8.4.1.1 Parameters of the AUDT APDU shall be supported for sending as specified in Table 4.7-18.

Table 4.7-18. AUDT Parameters when Sending

Prerequisite: ACNLS-UD-Sdr

	ACSE PDU parameter	ISO Status	ATN Support	Mnemonic
1	Protocol version	O	X	
2	Application Context Name	M	M	
3	Called AP title	O	X	
4	Called AE title	O	X	
5	Called AP invocation-identifier	O	X	
6	Called AE invocation-identifier	O	X	
7	Calling AP title	O	M	
8	Calling AE title	O	M	
9	Calling AP invocation-identifier	O	X	
10	Calling AE invocation-identifier	O	X	
11	Implementation information	O	X	
12	User information	M	M	
13	Authentication Mechanism-name	O	see text	
14	Authentication value	O	see text	

4.7.8.4.1.2 The AUDT parameters “Authentication mechanism-name” and “Authentication value” shall be supported for receiving, but are ignored if the Authentication functional unit is not supported by the receiver.

4.7.8.4.2 AUDT APDU receiver

4.7.8.4.2.1 Parameters of the AUDT APDU shall be supported for receiving as specified in Table 4.7-19.

Table 4.7-19. AUDT Parameters when Receiving

Prerequisite: ACNLS-UD-Rcv

	ACSE PDU parameter	**ISO Status**	**ATN Support**	**Mnemonic**
1	Protocol version	M	M	
2	Application Context Name	M	M	
3	Called AP title	O	M	
4	Called AE title	O	M	
5	Called AP invocation-identifier	O	X	
6	Called AE invocation-identifier	O	X	
7	Calling AP title	O	M	
8	Calling AE title	O	M	
9	Calling AP invocation-identifier	O	M	
10	Calling AE invocation-identifier	O	M	
11	Implementation information	O	X	
12	User information	M	M	
13	Authentication Mechanism-name	O	see text	
14	Authentication value	O	see text	

4.7.8.4.2.2 The AUDT parameters “Authentication mechanism-name” and “Authentication value” shall be supported, for sending, only if the Authentication functional unit is supported.

4.7.8.4.3 Supported parameter forms for the AUDT APDU shall be as specified in 4.6.6.3

Note.— There is a defect in ISO/IEC 10035-1/Amd.1 | ITU-T Rec. X.237/Amd.1 such that the tag value [12] is omitted from the calling-authentication-value element of the AUDT specification. It is assumed here that the tag value has been inserted.

4.7.8.5 Mapping to the Presentation Service

4.7.8.5.1 The mapping of the AUDT APDU and parameters to the presentation service primitives shall be performed by the CF as described in 4.7.4, which takes precedence over the direct mapping defined in clause 8 of ISO/IEC 10035-1 | ITU-T Rec. X.237 (1994).

Note.— The User Data of the P-UNIT-DATA service will always contain an AUDT APDU, therefore the requirements for full encoding in 4.3.2.6.2 do not apply.

4.8 SECURITY APPLICATION SERVICE OBJECT

4.8.1 Scope and Structure

Note 1.— The Security Application Service Object (ASO) defined here allows the ATN Upper Layers to provide security services to ATN applications. The Security ASO provides to its users the capability to agree on the set of security parameters (e.g. security algorithms and associated parameters, encryption keys) the security transformations and the security exchanges used to protect the data exchanged.

Note 2.— This section is structured as follows:

a) 4.8.1: SCOPE AND STRUCTURE contains the part's purpose and structure and a summary of the functions of the Security ASO.

b) 4.8.2: GENERAL REQUIREMENTS contains error processing requirements.

c) 4.8.3: THE SA ABSTRACT SERVICE contains the description of the abstract service provided by the Security ASO.

d) 4.8.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of the messages exchanged by Security ASOs using the Abstract Syntax Notation Number One (ASN.1).

e) 4.8.5: DEFINITION OF THE SECURITY ASO CONTROL FUNCTION (SA-CF) describes the Security ASO protocol in terms service primitives mapping as well as the exception handling procedures.

4.8.1.1 Security ASO Overview

Note 1.— The following types of services are provided to the Security ASO users (SA-users):

a) the Security ASO Start service to request the establishment of a secured dialogue and the exchange of the security parameters,

b) the Security ASO Send service to request protection of data.

Note 2.— The protection of data provided by the Security ASO mainly consists in appending to this data encrypted information which guarantees its origin and integrity.

Note 3.— The Security ASO makes use of the standard Security Exchange Service Element (SESE), which is specified in ISO/IEC 11586-2 | ITU-T Rec. X.831 (1995) (Service Definition) and ISO/IEC 11586-3 | ITU-T Rec. X.832 (1995) (Protocol Specification). It also makes use of a System Security Object (SSO), *as defined in ISO/IEC 10745 | ITU-T Rec. X.803 (1994). Details of the specific SSO are specified in Sub-Volume 8.*

*Note 4.— The **Security ASO Start** (SA-START) function.*

The SA-START function allows a SA-user to negotiate with the remote SA-user the set of security parameters. The actions performed by the systems supporting the SA-START function are the following:

a) *the initiating Security ASO formats and sends to the receiving Security ASO an **atnEstablish** security exchange item. This message contains:*

  1) *static security parameters: the security appendix type, the calling and the called peer entity identification,*

  2) *dynamic security parameters: a time field, and user data,*

  3) *the signature of the SA-user data to be computed with the security parameters,*

b) *the receiving Security ASO then passes the received data to the System Security Object (SSO), which :*

  1) *obtains the signing key certificate of the initiator, and, for ground systems, the Certificate Revocation List (CRL),*

  2) *checks that the signing key certificate of the initiator is not contained in the CRL,*

  3) *checks that the received signature is correct,*

  4) *checks, based on the time field contained in the message, that the request is still valid,*

  5) *computes a session key to be used to secure any subsequent communication with the peer.*

c) *the receiving Security ASO formats and sends to the initiating Security ASO an **atnEstablish** security exchange item containing:*

  1) *static security parameters: the security appendix type and associated parameters, the calling peer entity identification, and the called peer entity key agreement certificate,* *2) dynamic security parameters: a randomly generated number,*

3) the Message Authentication Code (MAC) of the SA-user data computed with the session key and the security parameters,

d) the initiating Security ASO then passes the security information to the SSO, which:

:

1) computes the session key to be used in any subsequent secure exchange,

2) checks that the received MAC value is correct.

Note 5.— The ***Security ASO Send*** *(SA-SEND) function*

The SA-SEND function allows peer SA-users to protect the data they exchange. This protection is based on the appending of a Message Authentication Code (MAC) to the data. The actions performed by the systems supporting the SA-SEND function are the following:

a) the sending Security ASO passes the data to the SSO, which computes an exchange key with the session key and security parameters,

b) the sending Security ASO formats and sends to the receiving Security ASO an ***atnProtectSign*** *security exchange item. This message contains:*

1) static security parameters: the security appendix type and associated parameters, the sending entity identification,

2) the MAC computed from the SA-user data computed with the exchange key and the security parameters, and

3) the SA-user data.

c) the receiving Security ASO then passes the message to the SSO, which computes the exchange key and checks that the received MAC value is correct

4.8.2 General Requirements

4.8.2.1 Error Processing Requirements

4.8.2.1.1 In the event of information input by the SA-user being incompatible with that able to be processed by the system, the SA-user shall be notified by local means.

4.8.2.1.2 In the event of an SA-user invoking a Security ASO service primitive when the Security ASO is not in a state specified in 4.8.5, the SA-user shall be notified by local means.

4.8.3 The SA Abstract Service

4.8.3.1 Service Description

Note 1.— The Security ASO service (SA service) is the abstract service which is used by a SA-user at its lower service boundary. There is no requirement to implement the SA service in any product. ATN end systems will in general be designed in such a way that it is impossible to detect (from external access) whether or not an interface corresponding to the SA service has been built.

Note 2.— 4.8.3 defines the abstract service interface for the SA service. The SA service is described in this section from the viewpoint of the SA-users, and the SA service provider.

Note 3.— This section defines the static behaviour (i.e, the format) of the SA service. Its dynamic behaviour (i.e., how it is used) is described in 4.8.5.

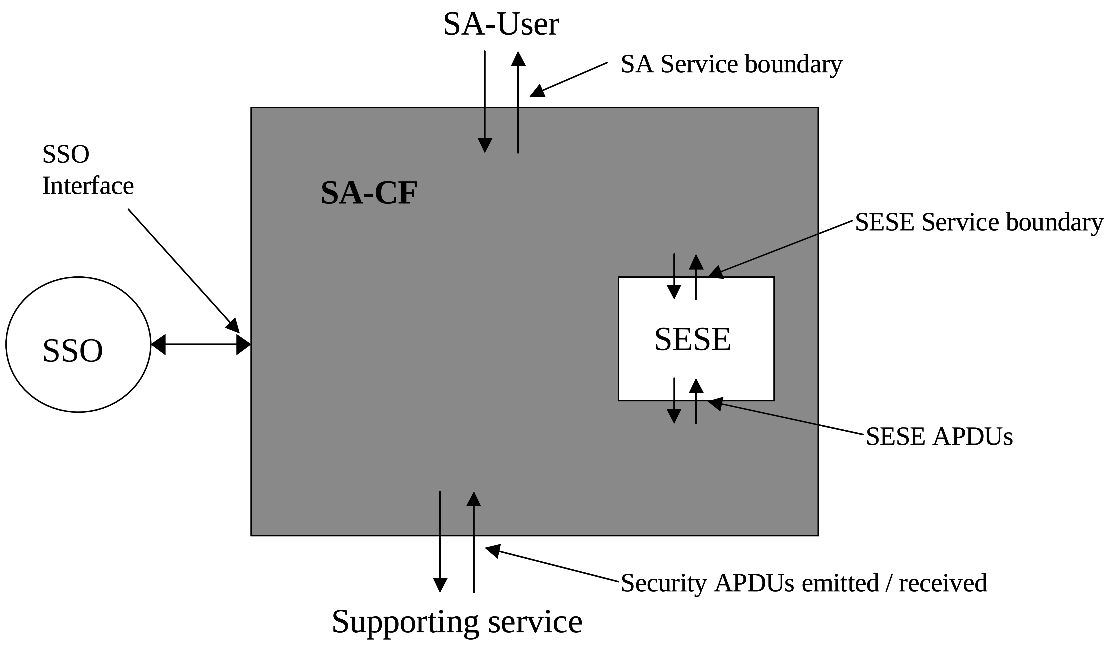


Figure 4.8-1. Functional model of the Security ASO

Note 4.— Figure 4.8-1 shows the functional model of the Security ASO. The functional modules identified in this model are the following:

a) *the SA-user,*

b) *the SA service boundary,*

c) *the Security ASO Control Function (SA-CF),* *d) the Security Exchange Service Element (SESE),*

e) the SESE service boundary,

f) the System Security Object (SSO),

g) the Supporting service, and

h) the SSO interface.

Note 5.— The SESE is the element in a communication system which executes the security specific protocol. In other words, it takes care of the SA service primitive sequencing actions, message creation, error and exception handling. The standard SESE is specified in ISO/IEC 11586-2 | ITU-T Rec. X.831 (1995) (Service Definition) and ISO/IEC 11586-3 | ITU-T Rec. X.832 (1995) (Protocol Specification).

Note 6.— The SSO is the local element in the communication system which performs the security functions or a combination of security functions (e.g. signature computation and check, generation of security variables such as session keys, time stamps and random values). This element does not perform any communication related function, and provides a local service which may be used by other components of the communication system. The SSO concept is defined in ISO/IEC 10745 | ITU-T Rec. X.803 (1994). Details of the specific ATN SSO are specified in 8.6.

Note 7.— The SESE interfaces only with the SA-CF. This SA-CF is responsible for mapping service primitives received from one element (such as the SESE, the SA-user, or the underlying communication service provider) to other elements which interface with it. The SA-CF maps the SESE APDUs transparently between the SESE lower interface and the Security ASO lower interface.

Note 8.— The supporting service is the boundary between the Security ASO and the underlying communication service provider.

Note 9.— There is no requirement to implement the service in a Security ASO product; however, it is necessary to implement the system in such a way that it will be impossible to detect (from the peer system) whether or not an interface has been built.

4.8.3.1.1 The SA abstract service shall consist of the set of the following services:

a) *SA-START* service as defined in 4.8.3.2,

b) *SA-SEND* service as defined in 4.8.3.3.

4.8.3.2 SA-START Service

Note.— The SA-START service is a confirmed service which is invoked by SA users both to negotiate the security parameters and to protect the exchanged data.

4.8.3.2.1 The SA-START service shall contain primitives and parameters as presented in Table 4.8-1.

Table 4.8-1. SA-START service parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Calling Entity ID	M	M(=)	M(=)	M(=)
Called Entity ID	M	M(=)	M(=)	M(=)
User Data	U		U	

Note 1.— Calling Entity ID

The initiating SA-user is identified by specifying either a name or an address in the SA-START request primitive. The SA-user therefore specifies a value for one, and only one, of the Calling Peer ID (with optional Sys-ID) and Calling Presentation Address parameters in the SA-START request primitive. In all SA-START invocations, the Calling Entity ID parameter identifies the entity that invoked the Request primitive.

The syntax of the Calling Entity ID parameter, is defined in 4.2.3.2.1, Note 4.

Note 2.— Called Entity ID

The SA-user identifies the intended peer SA-user by specifying either a name or an address in the SA-START request primitive. The SA-user therefore specifies a value for one, and only one, of the Called Peer ID (with optional Sys-ID) and Called Presentation Address parameters in the SA-START request primitive. In all SA-START invocations, the Called Entity ID identifies the entity to which the SA-START Request primitive was addressed.

The syntax of the Called Entity ID parameter is defined in 4.2.3.2.1, Note 3.

Note 3.— User Data

If present, the User Data parameter contains the information as provided by the SA-user.

4.8.3.3 SA-SEND Service

Note 1.— The SA-SEND service allows a SA-user to obtain a message authentication code, which is appended to the user data.

Note 2.— The SA-SEND service is an unconfirmed service which provides protection of data transferred between peer SA-users. The SA-START service must first have been successfully completed to establish the security relationship between the peers.

4.8.3.4 The SA-SEND service shall contain the primitives and parameters as presented in Table 4.8-2.

Table 4.8-2. SA-SEND service parameters

Parameter Name	**Req**	**Ind**
Local Entity ID	M	M(=)
Remote Entity ID	M	M(=)
User Data	U	U(=)

Note 1.— Local Entity ID.

The initiating SA-user is identified by specifying an AP-Title in the SA-SEND request primitive.

Note 2.— Called Entity ID

The SA-user identifies the intended peer SA-user by specifying an AP-Title in the SA-SEND request primitive.

Note 3.— User Data

The User Data parameter contains the information as provided by the SA-user.

4.8.4 Formal Definition of Messages

4.8.4.1 Formal definition of security exchange items

Note.— SESE APDUs consist of a parameterised part and a tailored part. The parameterised data type of the generic SESE APDUs is specified in clause 7 of ISO/IEC 11586-3 | ITU-T Rec X.832. It supports the definition of abstract syntaxes for tailored SESEs supporting any set of security exchanges. The tailored ATN-SESE APDUs are defined here.

4.8.4.1.1 The abstract syntax of the SESE protocol data units implemented in the Security ASO shall comply with the description contained in the ASN.1 module SecuredATNULs-Abstract-Syntax (conforming to ISO/IEC 8824-1 | ITU-T Rec X.680), as defined here.

Note 1.— The notation used in the following definitions (i.e. the SECURITY-EXCHANGE and SEC-EXCHG-ITEM constructs) is defined in ISO/IEC 11586-1 | ITU-T Rec X.830 (GULS). The ABSTRACT SYNTAX information object class is defined in ISO/IEC 8824-2 | ITU-T Rec X.681, Annex B.

Note 2.— The Object Identifiers atnPKI and securityExchanges are defined and assigned values in 9.2.1.3.

```
SecuredATNULs-Abstract-Syntax { iso(1) identified-organization(3) icao(27)
                atn-security-requirements(5) modules(1) abstract-syntax(2) }
DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- EXPORTS ALL --

IMPORTS
    -- From GULS --
    seseAPDUs FROM
        ObjectIdentifiers { joint-iso-ccitt genericULS(20) modules(1)
                objectIdentifiers(0) }

    SESEapdus, NoInvocationId FROM
        SeseAPDUs seseAPDUs

    -- From other ATN Security modules --
    securityExchanges FROM
        ATNObjectIdentifiers { iso(1) identified-organization(3)
                icao(27) atn(0) objectIdentifiers(0) }

    atnEstablishSE, atnProtectSignSE FROM
        ATNSecurityExchanges securityExchanges
```

```
;

securedATNULs-Abstract-Syntax ABSTRACT-SYNTAX ::=
{
        SESEapdus
        {
                {atnEstablishSE | atnProtectSignSE, ...},
                {NoInvocationId}
        } IDENTIFIED BY seseAPDUs
}
END

ATNSecurityExchanges { iso(1) identified-organization(3) icao(27)
                        atn-security-requirements(5) modules(1) securityExchanges(1) }
DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- EXPORTS ALL --

IMPORTS
        -- From other ISO/ITU-T standards --
        notation FROM
                ObjectIdentifiers { joint-iso-ccitt genericULS(20) modules(1)
                                objectIdentifiers(0) }

        AlgorithmIdentifier FROM
                AuthenticationFramework { joint-iso-ccitt ds(5) module(1)
                                        authenticationFramework(7) 3 }

        SECURITY-EXCHANGE {}, SEC-EXCHG-ITEM {} FROM
                Notation notation

        -- From other ATN Security modules --
        atnPKI FROM
                ATNObjectIdentifiers { iso(1) identified-organization(3)
                                icao(27) atn(0) objectIdentifiers(0) }      -- Defined in 9.2.1.3

        ATNCertificates, ATNSecurityDateTime, ECDSA-Sig-Value FROM
                ATN-PKI atnPKI                          -- Defined in Sub-Volume VIII
;

atnEstablishSE SECURITY-EXCHANGE ::= {
        SE-ITEMS       {atnEstablish}
        IDENTIFIER     local : 1 }

atnEstablish SEC-EXCHG-ITEM ::= {
        ITEM-TYPE      ATNEstablish
        ITEM-ID                  1 }

ATNEstablish ::= SEQUENCE {
        atnCertificates  SEQUENCE OF ATNCertificates OPTIONAL,
        atnSignature     ATNAppendix,
        ... }

atnProtectSignSE SECURITY-EXCHANGE ::= {
        SE-ITEMS       {atnProtectSign}
        IDENTIFIER     local : 2 }

atnProtectSign SEC-EXCHG-ITEM ::= {
        ITEM-TYPE      ATNProtectSign
        ITEM-ID                  1 }

ATNProtectSign ::= SEQUENCE {
        unprotected      OCTET STRING OPTIONAL,
        appendix         ATNAppendix,
        ... }

ATNAppendix ::= SEQUENCE {
        algorithmId      AlgorithmIdentifier OPTIONAL,
        validity         CHOICE {
                timeField        ATNSecurityDateTime ,
                random           INTEGER (0..4294967295),
                                 -- 32-bit unsigned integer
                counter          INTEGER (0..MAX),
                                 -- unsigned, unconstrained integer
                ...  } OPTIONAL,
        value            CHOICE {
                ecdsa-Signature          ECDSA-Sig-Value,
                hmac-Tag                 OCTET STRING (SIZE (4, ...)),
                ...  }
}

END --   End of ASN.1 module
```

4.8.5 Definition of the Security ASO Control Function (SA-CF)

4.8.5.1 Sequence Rules

4.8.5.1.1 With the exception of abort primitives, only the sequence of primitives illustrated in Figures 4.8-2 and 4.8-3 shall be permitted.

Note 1.— The following figures define the valid sequences of primitives that is possible to invoke during the operation of the Security ASO. They show the relationship in time between the service request and the resulting indication, and if applicable, the subsequent response and the resulting confirmation.

Note 2.— Primitives are processed in the order in which they are received.

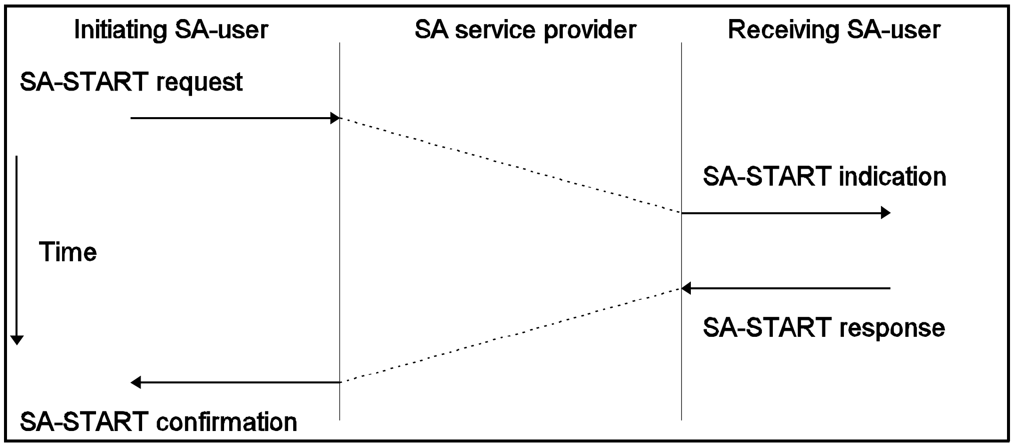


Figure 4.8-2. SA-START sequence diagram

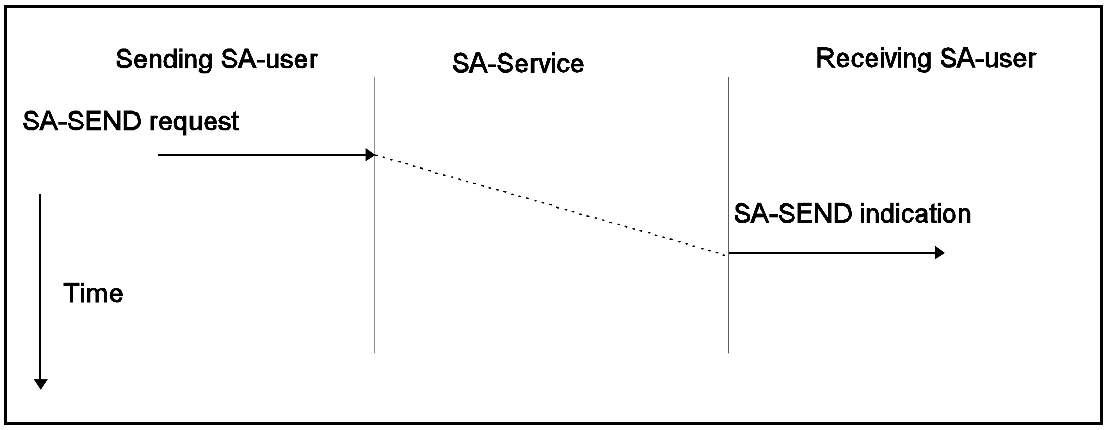


Figure 4.8-3 SA-SEND sequence diagram

4.8.5.2 Security ASO Function Description

4.8.5.2.1 Introduction

Note.— 4.8.5.2 presents requirements for the Security ASO Control Function (SA-CF) upon receipt of primitives at the Security ASO top and bottom interfaces and at the SESE top and bottom interfaces.

4.8.5.2.1.1 If an SSO function indicates to the SA-CF that it failed to perform a security operation requested by the SA-CF (e.g. insufficient resources, access to required information not available, signature check failure, etc.), then exception handling procedures, as described in 4.8.5.3 shall apply.

4.8.5.2.2 SA-CF Function Description

4.8.5.2.2.1 SA-START Request

4.8.5.2.2.1.1 Upon receipt of an SA-START Request primitive, the SA-CF shall:

a) retrieve, by locally-defined means, the appendix type to be used by the SSO for computing the appendix. Unless there is prior mutual agreement to the contrary, use the abstract value “Signature-appendix” as the type of the appendix to be used by the SSO.

b) if the Called EntityID refers to an airborne entity then get the signature key certificate path of the local system by activating the SSO-GetCertificatePath function with the following parameters:

Table 4.8-4.

SSO-GetCertificatePath function parameter	**ATN value**
Entity ID	Calling EntityID
Key Usage	abstract value “digitalSignature”

c) if the Called EntityID refers to an airborne entity then get the key agreement public key certificate path of the local system by activating the SSO-GetCertificatePath function with the following parameters:

Table 4.8-5.

SSO-GetCertificatePath function parameter	**ATN value**
Entity ID	Calling EntityID
Key Usage	abstract value “keyAgreement”

d) get the atnSignature by activating the SSO-Sign function with the following parameters:

Table 4.8-6.

SSO-Sign function parameter	**ATN value**
Source Peer	Calling EntityID
Destination Peer	Called EntityID
Appendix Type	set to the value retrieved in a)
User Data	SA-START Request *User Data* parameter

e) construct an atnEstablish security exchange item containing the following:

1) if the Called EntityID refers to an airborne entity, the signature key certificate retrieved in b) added in the *atnCertificates* field of the ATNEstablish, and

2) if the Called EntityID refers to an airborne entity, the key agreement public key certificate retrieved in c) added in the *atnCertificates* field of the ATNEstablish, and

3) the atnSignature retrieved in d) as the ATNEstablish *atnSignature* field.

f) invoke an SE-TRANSFER Request primitive with the following parameters:

Table 4.8-7.

SE-Transfer Request parameter	**ISO Status**	**ATN value**
Security exchange identifier	M	the atnEstablishSE object identifier
Invocation identifier	U	not used
Security exchange item	M	the atnEstablish security exchange item
Item identifier	U	abstract value that identifies an atnEstablish security exchange item
Start flag	U	"True"
End flag	U	not used

4.8.5.2.2.2 SE-TRANSFER Indication

4.8.5.2.2.2.1 Upon receipt of an SE-TRANSFER indication, if the SE-TRANSFER Indication *Security exchange identifier* parameter contains the atnEstablishSE object identifier, the SE-TRANSFER Indication *Item identifier* parameter has the abstract value that identifies an atnEstablish security exchange item, and the SE-TRANSFER Indication *Start flag* parameter has the abstract value *true*, the SA-CF shall:

a) retrieve the compressed certificate path of the remote system from the *atnCertificates* field of the ATNEstablish part of the SE-TRANSFER Indication *Security Exchange Item* parameter.

b) check the atnSignature is valid by activating the SSO-SignCheck function with the following parameters:

Table 4.8-8.

SSO-SignCheck function parameter	**ATN value**
Source Peer	Calling EntityID
Destination Peer	Called EntityID
Certificate Path	set to the value retrieved in a)
User Data	received User Data
Security Item	atnSignature field of the SE-TRANSFER Indication *Security Exchange Item* parameter.

c) if the atnSignature is valid then deliver a SA-START Indication containing the C*alling Entity ID* and *Called Entity ID* as the SA-START Indication *Calling Entity ID* and *Called Entity ID* parameter values.

4.8.5.2.2.2.2 Upon receipt of an SE-TRANSFER indication, if the SE-TRANSFER Indication *Security exchange identifier* parameter contains the atnEstablishSE object identifier, the SE-TRANSFER Indication *Item identifier* parameter has the abstract value that identifies an atnEstablish security exchange item, and the SE-TRANSFER Indication *Start flag* parameter has the abstract value *false*, the SA-CF shall:

a) extract the key-agreement-key compressed certificate path of the peer, from the *atnCertificates* element of the ATNEstablish field of the SE-TRANSFER Indication *Security Exchange Item* parameter, if present.

b) check the atnSignature is valid by activating the SSO-SignCheck function with the following parameters:

Table 4.8-9.

SSO-SignCheck function parameter	ATN value
Source Peer	Called EntityID
Destination Peer	Calling EntityID
Certificate Path	set to the value retrieved in a)
User Data	received User Data
Security Item	atnSignature field of the SE-TRANSFER Indication *Security Exchange Item* parameter.

c) if the atnSignature is valid then deliver a SA-START Confirmation containing the *Calling Entity ID* and *Called Entity ID* as the SA-START Confirmation *Calling Entity ID* and *Called Entity ID* parameter values.

4.8.5.2.2.2.3 Upon receipt of an SE-TRANSFER indication, if the SE-TRANSFER Indication *Security exchange identifier* parameter contains the atnProtectSignSE object identifier, the SE-TRANSFER Indication *Item identifier* parameter has the value that identifies an atnProtectSign security exchange item, and the SE-TRANSFER Indication *Start flag* parameter has the abstract value *true*, the SA-CF shall:

a) check the atnProtectSign item is valid by activating the SSO-ProtectSignCheck function with the following parameters:

Table 4.8-10.

SSO-ProtectSignCheck function parameter	ATN value
Source Peer	Remote EntityID
Destination Peer	Local EntityID
Security Item	atnProtectSign field of the SE-TRANSFER Indication *Security Exchange Item* parameter.

b) if the atnProtectSign item is valid then deliver a SA-SEND Indication containing the following fields:

1) *Remote Entity ID* as the SA-SEND Indication *Remote Entity ID* parameter value,

2) *Local Entity ID* as the SA-SEND Indication *Local Entity ID* parameter value, and

3) the *unprotected* element of the ATNProtectSign field of the SE-TRANSFER Indication *Security exchange item* parameter as the SA-SEND User Data parameter.

4.8.5.2.2.3 SA-START Response

4.8.5.2.2.3.1 Upon receipt of an SA-START Response primitive, the SA-CF shall:

a) retrieve, by locally-defined means, the appendix type to be used by the SSO. Unless there is prior agreement otherwise, use the abstract value "MAC-appendix" as the appendix type to be used by the SSO.

b) if the Called Entity ID refers to a ground entity, then get the key agreement public key certificate path of the local system by activating the SSO-GetCertificatePath function with the following parameters:

Table 4.8-11.

SSO-GetCertificatePath function parameter	**ATN value**
Entity ID	Called EntityID
Key Usage	abstract value keyAgreement

c) get the atnSignature by activating the SSO-Sign function with the following parameters:

Table 4.8-12.

SSO-Sign function parameter	**ATN value**
Source Peer	Called EntityID
Destination Peer	Calling EntityID
Appendix type	set to the value retrieved in a)
User Data	SA-START Response *User Data* parameter

d) construct an atnEstablish security exchange item containing the following:

1) the atnSignature retrieved in c) as the ATNEstablish *atnSignature* field,

2) the key agreement public key certificate retrieved in b) as the *atnCertificate* field of the ATNEstablish.

e) invoke an SE-TRANSFER Request primitive with the following parameters:

Table 4.8-13.

SE-Transfer Request parameter	**ISO Status**	**ATN value**
Security exchange identifier	M	the atnEstablishSE object identifier
Invocation identifier	U	not used
Security exchange item	M	the atnEstablish security exchange item
Item identifier	U	abstract value that identifies an atnEstablish security exchange item
Start flag	U	"False"
End flag	U	not used

4.8.5.2.2.4 SE-U-ABORT indication

4.8.5.2.2.4.1 Upon receipt of an SE-U-ABORT indication, if the SE-U-ABORT Indication *Item identifier* parameter has the abstract value that identifies an atnEstablish security exchange item, and the SE-TRANSFER Indication *Fatality indicator* parameter has the abstract value *fatal*, the SA-CF shall:

a) remove the security information by activating the SSO-Stop function with the following parameters:

Table 4.8-14.

SSO-Stop function parameter	**ATN value**
Calling Peer	Local EntityID
Called Entity ID	Remote EntityID

b) invoke by local means the dialogue service error handling procedures (see 4.3.3.1.2.4).

4.8.5.2.2.5 SE-P-ABORT indication

4.8.5.2.2.5.1 Upon receipt of an SE-P-ABORT indication, if the SE-P-ABORT Indication *Item identifier* parameter has the abstract value that identifies an atnEstablish security exchange item, and the SE-TRANSFER Indication *Fatality indicator* parameter has the abstract value *fatal*, the SA-CF shall:

a) remove the security information by activating the SSO-Stop function with the following parameters:

Table 4.8-15.

SSO-Stop function parameter	**ATN value**
Calling Peer	Calling EntityID
Called Peer	Called Entity ID

b) invoke by local means the dialogue service error handling procedures (see 4.3.3.1.2.4).

4.8.5.2.2.6 SA-SEND Request

4.8.5.2.2.6.1 Upon receipt of an SA-SEND Request primitive, the SA-CF shall:

a) retrieve, by locally-defined means, the appendix type to be used by the SSO for generating the appendix:

1) If no appendix type is locally-defined and if both the Local EntityID and the Remote EntityID refer to ground entities then, unless there is prior agreement otherwise, use the abstract value "Signature-appendix" as the appendix type to be used by the SSO.

2) If no appendix type is locally-defined and if either Local EntityID or Remote EntityID refers to an airborne entity then, unless there is prior agreement otherwise, use the abstract value "MAC-appendix" as the appendix type to be used by the SSO.

b) get the atnProtectSign by activating the SSO-ProtectSign function with the following parameters:

Table 4.8-16.

SSO-ProtectSign function parameter	**ATN value**
Source Peer	Local EntityID
Destination Peer	Remote EntityID
Appendix type	set to the value retrieved in a)

SSO-ProtectSign function parameter	**ATN value**
User Data	SA-SEND Request *User Data* parameter

c) construct an atnProtectSign security exchange item containing the *atnProtectSign* as ATNProtectSign field.

d) invoke an SE-TRANSFER Request primitive with the following parameters:

Table 4.8-17.

SE-Transfer Request parameter	**ISO Status**	**ATN value**
Security exchange identifier	M	the atnProtectSignSE object identifier
Invocation identifier	U	not used
Security exchange item	M	the atnProtectSign security exchange item
Item identifier	U	abstract value that identifies an atnProtectSign security exchange item
Start flag	U	"True"
End flag	U	not used

4.8.5.2.2.7 (Paragraph deleted)

4.8.5.2.2.8 APDU submitted by SESE

4.8.5.2.2.8.1 Upon receipt of an APDU from the SESE the SA-CF shall:

a) add the *Called* and *Calling Entity ID, or Local* and *Remote Entity ID* parameters, for correlation with the appropriate Dialogue Service primitives,

b) submit the APDU to the supporting service

Note.— In the present application context specification, the supporting service is always the Dialogue Service CF (see 4.3.3.8). The APDU will be either an SE-TRANSFER (SETR), SE-U-ABORT (SEAB) or SE-P-ABORT (SEPA) APDU.

4.8.5.2.2.9 APDU delivered by supporting service

4.8.5.2.2.9.1 Upon receipt of an APDU from the supporting service the SA-CF shall:

a) extract the *Called Entity ID, or the Calling Entity ID, the Local Entity ID, or the Remote Entity ID* and *User Data* parameters, and save them for later correlation with the output resulting from the SESE processing,

b) deliver the APDU to the SESE.

Note.— The APDU is expected to be either an SE-TRANSFER (SETR), SE-U-ABORT (SEAB) or SE-P-ABORT (SEPA) APDU.

4.8.5.3 Exception handling

4.8.5.3.1 Unrecoverable System Error

4.8.5.3.1.1 **Recommendation.—** *If a Security ASO has an unrecoverable system error, or in the case of a security failure (e.g. MAC incorrect. certificate invalid, etc.) the Security ASO should:*

a) if the Security ASO has an SESE exchange in progress, invoke SE-U-ABORT request with the following parameters:

Table 4.8-19.

SE-U-Abort Request parameter	**ISO Status**	**ATN value**
Invocation identifier	U	not used
Item identifier	U	not used
Error list	U	no error code
Fatality indicator	U	"Fatal"

b) behave as though an SE-U-ABORT indication has been received.

4.8.6 SESE Profile Requirements

Note.— The SESE requirements are described in many cases by means of completed protocol implementation conformance statement (PICS) proforma tables. In such tables, the "Ref." column contains a reference to the relevant section in the SESE PICS proforma ISO/IEC 11586-5 | ITU-T Rec. X.834.

4.8.6.1 Protocol details

4.8.6.1.1 The specification of the SESE protocol supported shall be as defined in Table 4.8-20.

Table 4.8-20. Identification of SESE Protocol Specification

Identification of Protocol Specification	ATN Support	Comments
ISO/IEC 11586-3:1996 | ITU-T Rec. X 832 (1995)	M	

4.8.6.2 Protocol mechanisms

4.8.6.2.1 Encoding

4.8.6.2.1.1 The system shall support that encoding which results from applying the ASN.1 packed encoding rules (basic, unaligned variant), as specified in ISO/IEC 8825-2 | ITU-T Rec. X.691 (1995), to the abstract syntax module specified in 4.8.4.

4.8.6.2.1.2 Packed encoding (basic, unaligned) shall be used for encoding all SESE Protocol Control Information (PCI) for interchange.

4.8.6.2.1.3 SESE APDUs shall be encoded as instances of the type SESEapdus specified in ISO/IEC 11586-3 | ITU-T Rec. X.832, including the encoding of the top level CHOICE.

4.8.6.2.1.4 Encoded SESE APDUs shall be treated as bit-oriented values that are not padded to an integral number of octets; the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

4.8.6.2.1.5 Padding bits shall be appended if necessary to achieve octet alignment of encoded values in the following cases only:

a) the 'unprotected' field in the ATNProtectSign type, which will be a bit-oriented ATN-App APDU value, padded to an integral number of octets,

b) the open ASN.1 type seItem in the SESEapdus.SETransfer APDU, since the PER standard requires, in clause 10.2, that this is encoded as an octet-aligned-bit-field, preceded by an encoded length in units of octets.

4.8.6.3 Supported APDUs

4.8.6.3.1 The SESE Protocol data units supported shall be as specified in Table 4.8-21.

Table 4.8-21. Supported SESE Protocol Data Units

Ref.	**APDU**	**Sender**		**Receiver**		**Comment**
		ISO Status	**ATN Support**	**ISO Status**	**ATN Support**	
A.8/ 1	SE-TRANSFER (SETR)	M	M	M	M	
A.8/ 2	SE-UABORT (SEAB)	M	M	M	M	
A.8/ 3	SE-P-ABORT (SEPA)	M	M	M	M	

4.8.6.3.2 Supported APDU parameters

4.8.6.3.2.1 SE-Transfer (SETR)

4.8.6.3.2.1.1 The parameters in the SETR APDU shall be supported as specified in Table 4.8-22.

Table 4.8-22. Supported SETR Parameters

		Sender		**Receiver**		**Comment**
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.9.1/1	SE Identifier	M	M	M	M	
A.9.1/2	Item Identifier	M	M	M	M	
A.9.1/3	SE Item	M	M	M	M	
A.9.1/4	Invocation Id (specify range supported)	O range	M	M range	M	see note
A.9.1/5	Start Flag	M	M	M	M	
A.9.1/15	End Flag	M	M	M	M	

Note.— The only value currently specified for the "Invocation Id" parameter is "NoInvocationId".

4.8.6.3.2.2 SE-U-Abort (SEAB)

4.8.6.3.2.2.1 The parameters in the SEAB APDU shall be supported as specified in Table 4.8-23.

Table 4.8-23. Supported SEAB Parameters

		Sender		**Receiver**		**Comment**
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.9.2/1	Invocation Id	C0	M	M	M	
A.9.2/2	Item Identifier	M	M	M	M	
A.9.2/3	Errors	M	M	M	M	

C0: if [A.9.1/4 supported] then M else N/A

4.8.6.3.2.3 SE-P-Abort (SEPA)

4.8.6.3.2.3.1 The parameters in the SEPA APDU shall be supported as specified in Table 4.8-24.

Table 4.8-24. Supported SEPA Parameters

		Sender		**Receiver**		**Comment**
Ref.	**Parameter**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.9.3/1	Invocation Id	C1	M	M	M	
A.9.3/2	Item Identifier	M	M	M	M	
A.9.3/3	Problem Code	M	M	M	M	

C1: if [A.9.1/4 supported] then M else N/A

4.8.6.3.2.3.2 The “Problem Code” parameter in the SEPA APDU shall be as specified in Table 4.8-25.

Table 4.8-25. Problem codes

Problem code		Sender		Receiver		Comment
Ref.	Parameter	ISO Status	ATN Support	ISO Status	ATN Support	
A.9.4/1	General problem code	M	M	M	M	
A.9.4/2	Transfer problem code	M	M	M	M	
A.9.4/3	Abort problem code	M	M	M	M	

4.8.6.4 Abstract syntax

4.8.6.4.1 The supported abstract syntax shall be as defined in Table 4.8-26.

Table 4.8-26. Identification of the supported abstract syntax

Abstract syntax name	Comments
secids modules (1) abstract-syntax (2)	The root OID "secids" is defined in Sub-Volume 9 as icao-atn-security-requirements.

4.8.6.5 Application Context

4.8.6.5.1 The supported application context shall be as defined in Table 4.8-27.

Table 4.8-27. Application Context

Ref.		ISO Status	ATN Support	Comment
A.11/1	Basic SESE Application Context	O		
A.11/2	Object identifier for ATN application context		M	

4.8.6.6 Security exchanges

4.8.6.6.1 The supported security exchanges and security exchange items are defined in 4.8.4 and are listed in this chapter.

4.8.6.6.2 The supported classes of security exchanges shall be as defined in Table 4.8-28.

Table 4.8-28. Class of Security Exchange Supported

Ref.		**ISO Status**	**ATN Support**	**Comment**
A.12.1/1	Alternating class of security exchanges	O	M	
A.12.1/2	Arbitrary class of security exchanges	O	M	

4.8.6.6.3 The supported security exchanges shall be as defined in Table 4.8-29.

Table 4.8-29. Supported Security Exchanges

Ref.		**ISO Status**	**ATN Support**	**Comment**
A.12.2/1	Directory Authentication Exchange (one way)	C2	n/a	
A.12.2/2	Directory Authentication Exchange (two ways)	C2	n/a	
A.12.2/3	Simple Negotiation Exchange	C2	n/a	
A.12.2/4.1	atnEstablishSE (1)		M	
A.12.2/4.2	atnProtectSignSE (2)		M	

C2: if [A.12.1/1 supported] then O else N/A

4.8.6.6.4 Directory Authentication Exchange (one way)

4.8.6.6.4.1 The security exchange item “Directory Authentication Exchange (one way)” shall be supported as specified in Table 4.8-30.

Table 4.8-30. Directory Authentication Exchange (one way)

		Sender		**Receiver**		**Comment**
Ref.	**Security Exchange Item**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.12.3/1	Credentials	C3	n/a	C3	n/a	

C3: if [A.12.2/1 supported] then M else N/A

4.8.6.6.5 Directory Authentication Exchange (two ways)

4.8.6.6.5.1 The security exchange item "Directory Authentication Exchange (two ways)" shall be supported as specified in Table 4.8-31.

Table 4.8-31. Directory Authentication Exchange (two ways)

		Sender		**Receiver**		**Comment**
Ref.	**Security Exchange Item**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.12.4/1	Credentials	C4	n/a	C4	n/a	
A.12.4/2	Responder credentials	C4	n/a	C4	n/a	
A.12.4/3	Authentication failure	C4	n/a	C4	n/a	

C4: if [A.12.2/2 supported] then M else N/A

4.8.6.6.6 Simple Negotiation Exchange

4.8.6.6.6.1 The security exchange item "Simple Negotiation Exchange" shall be supported as specified in Table 4.8-32.

Table 4.8-32. Simple Negotiation Exchange

		Sender		**Receiver**		**Comment**
Ref.	**Security Exchange Item**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.12.5/1	Offered Ids	C5	n/a	C5	n/a	
A.12.4/2	Accepted Ids	C5	n/a	C5	n/a	

C5: if [A.12.2/3 supported] then M else N/A

4.8.6.6.7 The security exchange atnEstablishSE shall be supported as specified in Table 4.8-33.

Table 4.8-33.

		Sender		Receiver		Comment
Ref.	**Security Exchange Item**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.12.2/4.1	atnEstablish		M		M	

4.8.6.6.8 The security exchange atnProtectSignSE shall be supported as specified in Table 4.8-34.

Table 4.8-34.

		Sender		Receiver		Comment
Ref.	**Security Exchange Item**	**ISO Status**	**ATN Support**	**ISO Status**	**ATN Support**	
A.12.2/4.2	atnProtectSign		M		M	

4.9 GENERIC ATN COMMUNICATIONS SERVICE SPECIFICATION

4.9.1 Scope and Structure

Note 1.— The Generic ATN Communications Service (GACS) defined here provides a service which allows a user of the service to transfer data transparently across the ATN to another user (or to multiple users). The user is able to specify the required quality of service (QoS) and recipient addressing parameters separately for each data transfer.

Note 2.— This specification is designed to optimise the use of communications bandwidth, and consequently uses the Dialogue Service defined in 4.2 and 4.7.2, including extensions for unit data and presentation address handling services. The Dialogue Service in turn uses the ATN Transport service defined in 5.5.

Note 3.— The GACS user is able to select the required level of service, which in turn results in the use of either a connection-oriented (CO) or connectionless (CL) supporting protocol stack.

Note 4.— This section is structured as follows:

a) 4.9.1: SCOPE AND STRUCTURE contains the purpose and structure of the GACS Specification, and a background to the functionality defined herein. Two possible architectures for implementing the GACS protocol are outlined.

b) 4.9.2: GACS SERVICE DEFINITION specifies the abstract service provided by the GACS specification.

c) 4.9.3: PROTOCOL DEFINITION defines the protocol data units (PDUs) exchanged by GACS entities, using Abstract Syntax Notation One (ASN.1), and describes the sequences of events allowed by the GACS protocol.

d) 4.9.4: COMMUNICATION REQUIREMENTS contains the requirements that the GACS ASE application imposes on the underlying communication system.

e) 4.9.5: USER REQUIREMENTS outlines the requirements that a user of a GACS ASE must meet.

f) 4.9.6: SUBSETTING RULES contains the conformance requirements which all implementations of the GACS protocol obey.

4.9.1.1 Generic ATN Communications Service Overview

Note 1.— The service defined here provides a generic data transfer capability. To allow maximum flexibility and optimal use of data link bandwidth, the following basic services are defined;

a) G-TRANSFER,

b) G-TRANSFER-CONFIRMED,

c) G-END,

d) G-MULTICAST.

Note 2.— The G-TRANSFER service

The G-TRANSFER service allows a user to transfer data to a specified destination (or multiple destinations) via the ATN internet. A number of options are defined:

a) Connectionless Mode. If the user does not require a resilient communications service (e.g. because the data transfer is not mission-critical, or because the user application itself implements an error recovery protocol) then this can be requested per transfer. In this case, a connectionless (CL) protocol stack, if available, will be used to transfer the data, provided the size constraints of the CL stack are not exceeded.

b) Connection-Oriented Mode. If the user does require a resilient communications service (e.g. because the data transfer is mission-critical, and the user application itself does not implement an error recovery protocol) then this can be requested per transfer. In this case, a connection-oriented (CO) protocol stack will be used.

c) Multi-shot Option. If the user intends to perform multiple data transfers with the same Quality of Service (QoS) requirements and the same destination(s), then it can optionally request a "multi-shot" mode. This establishes and maintains a two-way communication relationship with the specified peer(s), and provides an optimised use of the communications link, using a CO protocol stack. In “multi-shot” mode, G-TRANSFER requests in both directions between peer GACS-Users use the same established dialogue if possible.

Note 3.— The G-TRANSFER-CONFIRMED service

The G-TRANSFER-CONFIRMED service allows a user to transfer data to a specified destination (or multiple destinations) and to receive confirmation that the data was delivered to the remote user application(s).

The confirmed service supports the same resilience and multi-shot options as the G-TRANSFER service.

Note 4.— The G-END service

The G-END service is an optional service which allows a user of the multi-shot option to inform the GACS service that a communications relationship with the specified peer(s) is no longer required to be maintained. This allows an orderly freeing of resources, and an assurance that there is no data in transit to or from that particular peer(s).

If G-END is not used, then any established communications relationship between two peers will automatically be ended by the GACS service on expiry of a configurable inactivity timer.

Note 5.— The G-MULTICAST service

The G-MULTICAST service is an optional local service which provides users with access to the multicast capabilities of the connectionless transport service, where such capabilities exist, so that a user can receive data items addressed to a group address.

Note 6.— In the G-TRANSFER and G-TRANSFER-CONFIRMED services, data items can optionally be sent to a specified Presentation address rather than to a named peer. This provision is only supported if the appropriate Upper Layer naming and addressing enhancements to the Dialogue service are also supported.

4.9.1.2 GACS Realisation

Note.— The GACS service provision can be realised alternatively as an "Application Layer data transfer protocol" or as a "simple generic service". The two approaches are very different and have different fields of applicability.

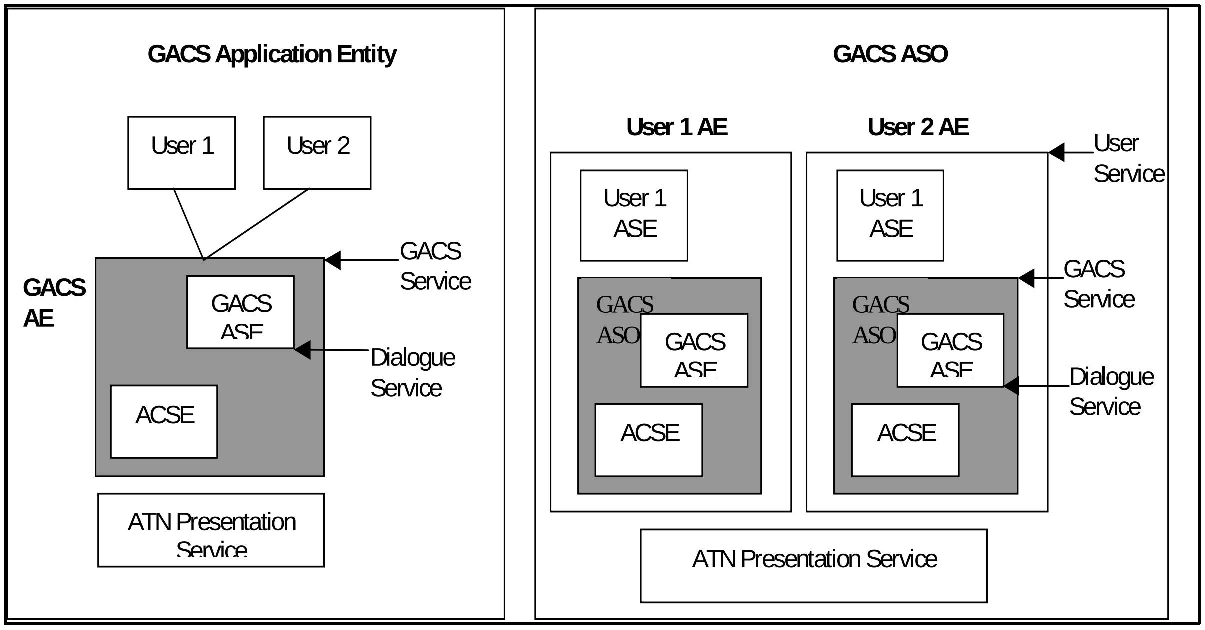


Figure 4.9-1. GACS Application versus GACS ASO

4.9.1.2.1 GACS implementations shall exhibit external behaviour consistent with one of the two alternative architectures illustrated in Figure 4.9-1, namely:

a) GACS can be realised as an ATN Application Entity (AE) providing an ATN access point to existing (e.g. ACARS-based) and future applications which are not specified to use the defined ATN upper layer architecture.

b) GACS can be realised as an ATN Application Service Object (ASO) or "internal service" providing an enhanced dialogue service in the ATN Upper Layers Architecture to future air-ground and ground-ground ATN applications (ATC and AOC).

Note.— The GACS AE approach is appropriate for the migration of existing applications. However, this approach is not the only approach for applications to use the ATN. The GACS ASO (enhanced dialogue service) approach would be preferred for any new ATC or AOC application.

4.9.1.3 The GACS Application Entity

4.9.1.3.1 When GACS is realised as an ATN AE, then the following provisions shall be satisfied:

a) An ATN address is allocated to the GACS AE.

b) The GACS application is identified by an Application Entity Title whose value is an OBJECT IDENTIFIER such as: {iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) <end-system-id> (n) ops (0) gacs (12) <Sys-ID> (k)}, as defined in 4.3.2.3.5.

c) The CM Application (2.1) is used to exchange the address and version number of the GACS application (app-type = GAC (12)) in the air-ground ATSC environment.

Note 1.— This would be a distinct ATN application installed in aircraft and ground systems acting as a point of access to the ATN.

Note 2.— It would be natural to implement the GACS service as an application programming interface (API), providing a communications interface to user applications.

Note 3.— Several GACS-Users could use services from the same GACS Application. The GACS Application therefore multiplexes data supplied by GACS-Users over the same dialogue when the intended recipient and the requested communication characteristics are identical.

Note 4.— GACS-Users in this approach are not considered as fully integrated ATN applications; they have no distinct ATN names and no ATN addresses. CM is not used to exchange the version numbers of the users, only of the GACS Application itself. Specific mechanisms would be implemented to switch incoming data to the relevant GACS-User, based on the message-type field.

Note 5.— The GACS Application itself does not know anything about the contents of the User Data, or the encoding rules for these user-defined data structures. Typical communication functions, such as sequence numbering and request/reply correlation are entirely the responsibility of the GACS-User applications.

4.9.1.4 The GACS Application Service Object

4.9.1.4.1 When the GACS protocol is realised as an ASO, then the following provisions shall be satisfied:

a) The AE in which the GACS protocol is embedded is considered as a fully integrated ATN application identified by a specific ATN address.

b) The AE is identified by an Application Entity Title whose value is an OBJECT IDENTIFIER such as: {iso (1) identified-organisation (3) icao (27) atn-end-system-air (1) <end-system-id> (n) ops (0) <app-type> (l) <Sys-ID> (k)}, as defined in 4.3.2.3.5.

c) If the CM Application (2.1) is used to exchange the address and version number specific to the application type of the AE in the air-ground ATSC environment, then the <app-type> value is registered in 4.3.2.3.4.

Note 1.— The GACS ASO is defined to provide new ASEs with an enhanced service within the AE. New ASEs can be developed over either the Dialogue service or the GACS service, depending upon their requirements.

Note 2.— GACS itself is not identified as an ATN application in this configuration. CM is used to assess the application capability of the peer system and to exchange dynamically the addressing information of these applications in the air-ground environment.

Note 3.— This architecture is completely in line with the ATN ULA specified in 4.1.

Note 4.— ASEs can be privately defined. The ASE protocol and the format of the data exchanged by AOC ASEs for example does not need to be standardised by ICAO or disclosed externally. These ASEs are "black boxes" for the ATN (the same way the existing ASEs such as ADS, CPDLC, etc. are considered as "black boxes" by the ULCS architecture).

4.9.2 GACS Service Definition

4.9.2.1 Service Primitives

4.9.2.1.1 Implementations which claim to support the GACS functionality shall exhibit the behaviour defined by the service primitives in Table 4.9.2-1.

Note.— There is no requirement to implement the GACS service interface in any implementation; however, it is necessary to implement the end system in such a way that it will be impossible to detect (from the peer system) whether or not an interface has been built.

Table 4.9.2-1. Summary of GACS Service primitives

Service	**Description**
G-TRANSFER	This is an unconfirmed service used to transfer User-Data between communicating GACS-Users.
G-TRANSFER-CONFIRMED	This is a confirmed service used to transfer User-Data between communicating GACS-Users, and to provide the sender with confirmation that the data was received at the remote peer system(s).
G-END	This is an unconfirmed service used optionally to terminate an established communications relationship between communicating GACS-Users.
G-MULTICAST	This is an unconfirmed service used optionally to indicate whether a user wishes to receive data sent to a particular group address.

4.9.2.2 Sequence of Primitives

4.9.2.2.1 Implementations which claim to support the GACS functionality shall exhibit behaviour allowing communicating users, consistent with the appropriate use of the corresponding service primitives, to:

a) send and receive user data;

b) optionally, establish a communications relationship between peer users for performing multiple data transfers;

c) optionally, terminate an established communications relationship.

4.9.2.2.2 The service shall permit a user to send and receive data at any time by using the G-TRANSFER or G-TRANSFER-CONFIRMED service.

4.9.2.2.3 The service shall permit a user to invoke the G-END service at any time after the multi-shot mode has been selected, and before the multi-shot inactivity timer has expired.

4.9.2.2.4 If a G-END request is invoked when there is no corresponding established communications relationship, an error indication shall be returned to the invoker.

Note.— The G-END service may only be used successfully after a multi-shot communications relationship has been established by using the appropriate option of the G-TRANSFER (-CONFIRMED) service.

4.9.2.2.5 The service shall permit a user to invoke the G-MULTICAST service at any time.

4.9.2.2.6 If a G-MULTICAST request is invoked when there is no supporting connectionless multicast service available, an error indication shall be returned to the invoker.

4.9.2.3 The G-TRANSFER service

4.9.2.3.1 The behaviour defined by the G-TRANSFER service shall be provided to enable the transparent transmission of data between GACS-Users.

Note 1.— G-TRANSFER is an unconfirmed service which is invoked by one GACS-User (the initiator) to send data to a peer GACS-User (or multiple peer users). If more than one recipient is specified, this is treated as multiple sequential single-recipient invocations of the service. G-TRANSFER request and indication service primitives are defined, as illustrated in Figure 4.9.2-1.

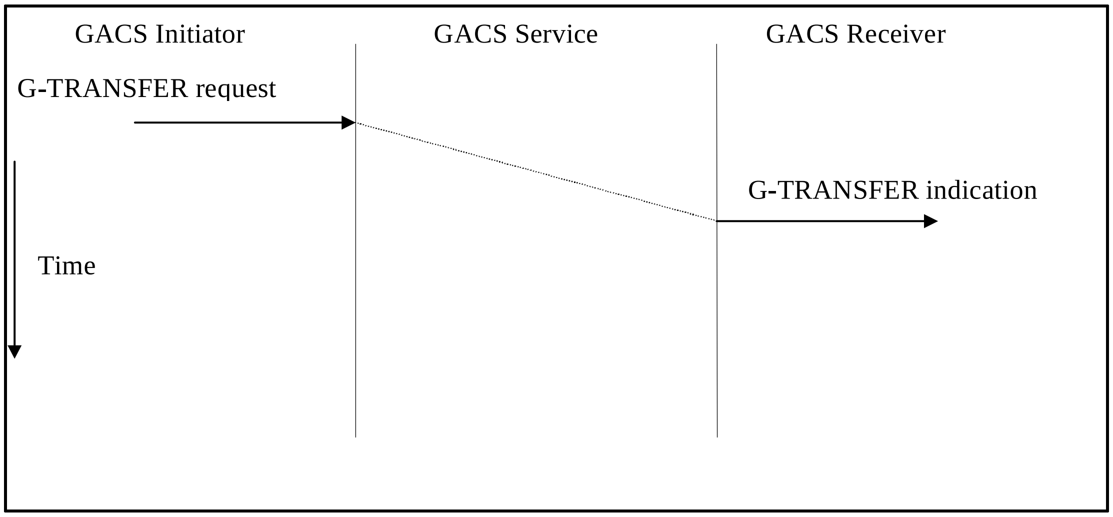


Figure 4.9.2-1. G-TRANSFER sequence diagram

Note 2.— The initiating GACS-User issues a G-TRANSFER request primitive. When the receiving GACS-User receives the G-TRANSFER indication primitive, the User Data is presented transparently to that user. It is a local matter to decide whether or not any reply is needed. Either GACS-User may issue a G-TRANSFER request at any time. Any sequencing constraints must be enforced by the GACS-Users themselves. The parameters of the G-TRANSFER primitives are specified in Table 4.9.2-2.

Table 4.9.2-2. G-TRANSFER parameters

Parameter Name	***Req***	***Ind***
Recipient List	*M*	
Sender	*U*	*C(=)*
Message Type	*U*	*C(=)*
Message Identifier	*U*	*C(=)*
Message Reference	*U*	*C(=)*
GACS-User Version Number	*U*	*C(=)*
Security Requirements	*U*	*C(=)*
Class of Communication	*M*	*M(=)*
Priority	*M*	*M(=)*
RER	*U*	*C(=)*
Requested Level of Service	*M*	*M(=)*
User Data	*U*	*C(=)*

Note 3.— The Recipient List parameter is used to specify with maximum flexibility the name or address of the location of the intended peer GACS-User(s). It is a list of one or more elements. Each element takes an abstract value corresponding to either a Peer ID (i.e. 24-bit ICAO aircraft-id for an airborne location, or ICAO facility designator for a registered ground location) plus optional system identifier, or a PSAP address (for any location).

Note 4.— The Sender parameter is optionally used to request that the recipient(s) be informed of the location of the initiating GACS-User. It takes an abstract value corresponding to either a Peer ID format plus optional system identifier, or a PSAP address. Its presence in the indication primitive is conditional upon it being specified by the GACS-User in the request primitive.

Note 5.— The Message Type parameter allows peer GACS-Users to refer unambiguously to a defined abstract syntax. For example, it could be used to indicate that the User Data conforms to a format defined by an external organisation, i.e. it belongs to a well-defined message set. The Message Type identifies the message set, and is similar to a protocol identifier. It could be used for routing data of a given type to an appropriate element of the overall GACS-User functionality. The use and permitted values of this parameter are out of the scope of the GACS service specification, and will be defined in GACS-User specifications. Its presence in the indication primitive is conditional upon it being specified by the GACS-User in the request primitive.

Note 6.— The Message Identifier parameter allows a GACS-User to assign an identifier to this particular data transfer. The parameter is optional in the request primitive. Its presence in the indication primitive is conditional upon it being specified by the initiating GACS-User in the request primitive.

Note 7.— The Message Reference parameter allows peer GACS-Users to refer unambiguously to a previous data transfer, by setting it equal to the Message Identifier used in that transfer. The parameter is optional in the request primitive. Its presence in the indication primitive is conditional upon it being specified by the initiating User in the request primitive.

Note 8.— The Class of Communication parameter requires the initiating GACS-User to specify the traffic type and routing requirements for this data transfer. Valid abstract values for ATSC, AOC, ATN Administrative Communications, General Communications and ATN Systems Management Communications are defined in the "Semantics" column of Table 5.6-1.

Note 9.— The Priority parameter requires the initiating GACS-User to specify in the request primitive its requirements for the priority of this data transfer. Valid abstract values are defined in the "Message Categories" column of Table 1-2.

Note 10.— The Requested Level of Service parameter allows the Initiating GACS-User to specify its requirements for the integrity and style of use of the communications channel. If the GACS-User intends to perform multiple data transfers with the same Quality of Service (QoS) requirements to the same destination(s), then it can optionally specify a "Multi-shot" mode. Valid abstract values are:

a) Single shot, no error recovery, unconfirmed service,

b) Single shot, error recovery, unconfirmed service,

c) Multi shot, error recovery, unconfirmed service.

Note 11.— The User Data parameter allows the Initiating-GACS-User to send User Data transparently to the Receiving GACS-User(s). Its presence in the indication primitive is conditional upon it being specified by the GACS-User in the request primitive. The maximum length of the User Data will typically be an implementation constraint and must be defined in the specification of the applications which use the GACS service.

Note 12.— The GACS-User Version Number, RER and Security Requirements parameters are exactly as defined for the Dialogue Service in 4.2.

4.9.2.4 The G-TRANSFER-CONFIRMED service

4.9.2.4.1 The behaviour defined by the G-TRANSFER-CONFIRMED service shall be provided to enable the transparent transmission of data between GACS-Users, with confirmation of data delivery to the recipient system(s).

Note 1.— G-TRANSFER-CONFIRMED is a confirmed service which is invoked by one GACS-User (the initiator) to send data to one or more peer GACS-User(s). If more than one recipient is specified, this is treated as multiple sequential single-recipient invocations of the service. G-TRANSFER-CONFIRMED request, indication and confirmation service primitives are defined, as illustrated in Figure 4.9.2-2.

Note 2.— The initiating GACS-User issues a G-TRANSFER-CONFIRMED request primitive. When the receiving GACS-User receives the G-TRANSFER-CONFIRMED indication primitive, the User Data is presented transparently to that user, and a G-TRANSFER-CONFIRMED confirmation primitive is automatically returned to the initiator. It is a local matter to decide whether or not any user reply to the indication primitive is needed. Either GACS-User may issue a G-TRANSFER-CONFIRMED request at any time. Any sequencing constraints must be enforced by the GACS-Users themselves. The parameters of the G-TRANSFER-CONFIRMED primitives are specified in Table 4.9.2-3.

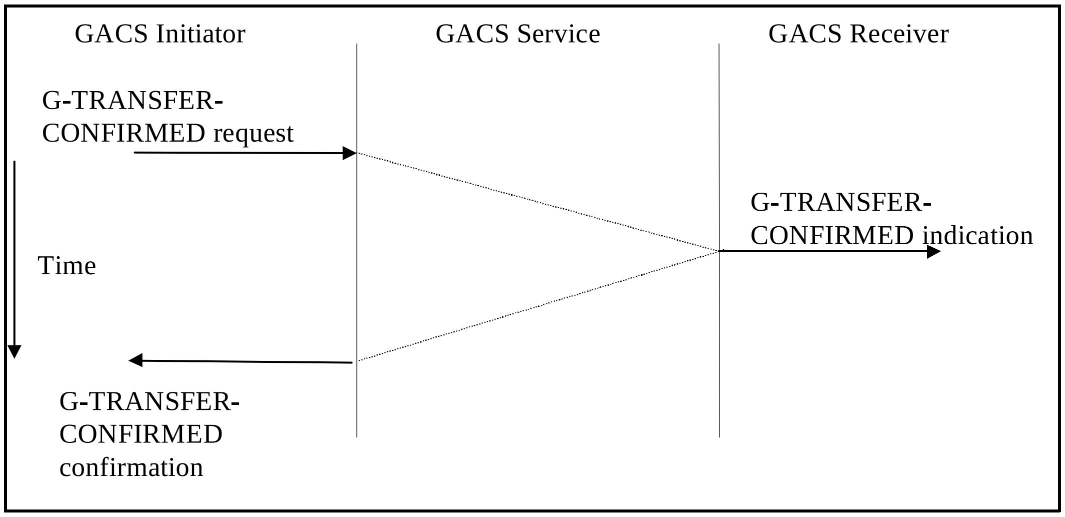

GACS Initiator
GACS Service
GACS Receiver
G-TRANSFER-CONFIRMED request
G-TRANSFER-CONFIRMED indication
Time
G-TRANSFER-CONFIRMED confirmation

Figure 4.9.2-2. G-TRANSFER-CONFIRMED sequence diagram

Table 4.9.2-3. G-TRANSFER-CONFIRMED primitive parameters

Parameter Name	***Req***	***Ind***	***Cnf***
Recipient List	*M*		
Sender	*U*	*C(=)*	*M*
Message Type	*U*	*C(=)*	*C(=)*
Message Identifier	*M*	*M(=)*	
Message Reference	*U*	*C(=)*	*M*
GACS-User Version Number	*U*	*C(=)*	*C*
Security Requirements	*U*	*C(=)*	
Class of Communication	*M*	*M(=)*	

Parameter Name	***Req***	***Ind***	***Cnf***
Priority	*M*	*M(=)*	
RER	*U*	*C(=)*	
Requested Level of Service	*M*	*M(=)*	
Result			*M*
User Data	*U*	*C(=)*	

Note 3.— The Recipient List parameter is identical to the G-TRANSFER service.

Note 4.— The Sender parameter is identical to the G-TRANSFER service, except that it is additionally used in the confirmation primitive to convey the identity of the location which invoked the confirmation. Its presence in the confirmation primitive is mandatory.

Note 5.— The Message Type parameter is identical to the G-TRANSFER service except that its presence in the confirmation primitive is conditional upon it being specified by the GACS-User in the request primitive.

Note 6.— The Message Identifier parameter is used to allow initiating GACS-Users to correlate confirmation primitives with previously invoked request primitives. The parameter is therefore mandatory in the request primitive. The value in the indication primitive is equal to the value set by the initiating GACS-User in the request primitive.

Note 7.— The Message Reference parameter is identical to the G-TRANSFER service except that it is mandatory in the confirmation primitive, where its value is equal to the Message Identifier value used in the data transfer which is being confirmed.

Note 8.— The Class of Communication parameter is identical to the G-TRANSFER service.

Note 9.— The Priority parameter is identical to the G-TRANSFER service.

Note 10.— The Requested Level of Service parameter allows the Initiating User to specify its requirements for the integrity of the communications channel. Valid abstract values are:

a) Single shot, no error recovery, confirmed service,

b) Single shot, error recovery, confirmed service,

c) Multi-shot, error recovery, confirmed service.

Note 11.— The GACS-User Version Number is as defined for the Dialogue Service in 4.2. Its presence in the indication and confirmation primitives is conditional upon it being specified by the initiating GACS-User in the request primitive. The value in the confirmation primitive is set by the GACS-Provider in the peer system, if possible, based on local knowledge of the peer GACS-User(s).

Note 12.— The RER and Security Requirements parameters are exactly as defined for the Dialogue Service in 4.2.

Note 13.— The Result parameter informs the Initiating GACS-User of the outcome of the G-TRANSFER-CONFIRMED request. Valid abstract values are:

a) Successful delivery,

b) Delivery failed,

c) Service problem - delivery status uncertain.

Note 14.— The User Data parameter is identical to the G-TRANSFER service.

4.9.2.5 The G-END service

4.9.2.5.1 The behaviour defined by the G-END service shall be provided to enable the orderly termination of a communications relationship between GACS-Users.

Note 1.— G-END is an unconfirmed service which is optionally invoked by one GACS-User (who is then the initiator) to terminate a communications relationship with one or more peer GACS-User(s). If more than one recipient is specified, this is treated as multiple sequential single-recipient invocations of the service. G-END request and indication service primitives are defined, as illustrated in Figure 4.9.2-3. Implementations that provide an exposed GACS service interface may choose not to provide the G-END indication, as this is purely informative.

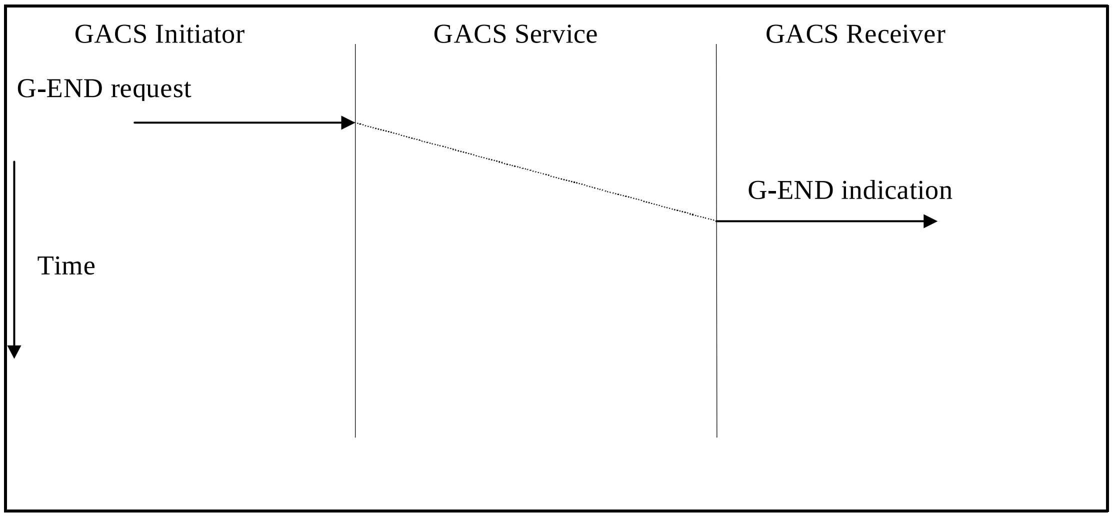


Figure 4.9.2-3. G-END sequence diagram

Note 2.— The initiating GACS-User issues a G-END request primitive at any time after using the G-TRANSFER or G-TRANSFER-CONFIRMED service with a multi-shot Level of Service. When the receiving GACS-User receives the G-END indication primitive, it knows that the current communications relationship with the peer is over. A new relationship may be established at any time. It is a local matter to decide whether or not any user reply is needed. Any sequencing constraints must be enforced by the GACS-Users themselves. The parameters of the G-END primitives are specified in Table 4.9.2-4.

Table 4.9.2-4. G-END parameters

Parameter Name	***Req***	***Ind***
Recipient List	*M*	
Sender		*M*
Message Type	*U*	*C(=)*
Message Reference	*U*	*C(=)*

Note 3.— Possible collisions of the G-END primitives are handled internally by the Dialogue Service, and are not visible to GACS-Users.

Note 4.— If there is more than one dialogue open with the specified peer (e.g. multiple dialogues with different QoS parameters), then the mechanism for associating the G-END request with the correct dialogue is a local implementation matter.

Note 5.— The Recipient List parameter is identical to the G-TRANSFER service.

Note 6.— The Sender parameter is used to convey the identity of the location which invoked the D-END request primitive. This is required in cases where a GACS-User has set up multi-shot connections to multiple peers, and it needs to identify which peer has invoked the D-END service. Its presence in the indication primitive is therefore mandatory. The syntax is the same as the Sender parameter in the G-TRANSFER service.

Note 7.— The Message Type parameter is identical to the G-TRANSFER service.

Note 8.— The Message Reference parameter is identical to the G-TRANSFER service .

4.9.2.6 The G-MULTICAST service

4.9.2.6.1 The behaviour defined by the G-MULTICAST service shall be provided to enable or disable the receipt of data addressed to a group address.

Note 1.— G-MULTICAST is an unconfirmed service which is optionally invoked by a GACS-User to inform the local communications system whether that user wishes to receive data sent to a particular group address. This service is only available when supported by a connectionless communications provider.

Note 2.— A G-MULTICAST request primitive is defined, as illustrated in Figure 4.9.2-4.

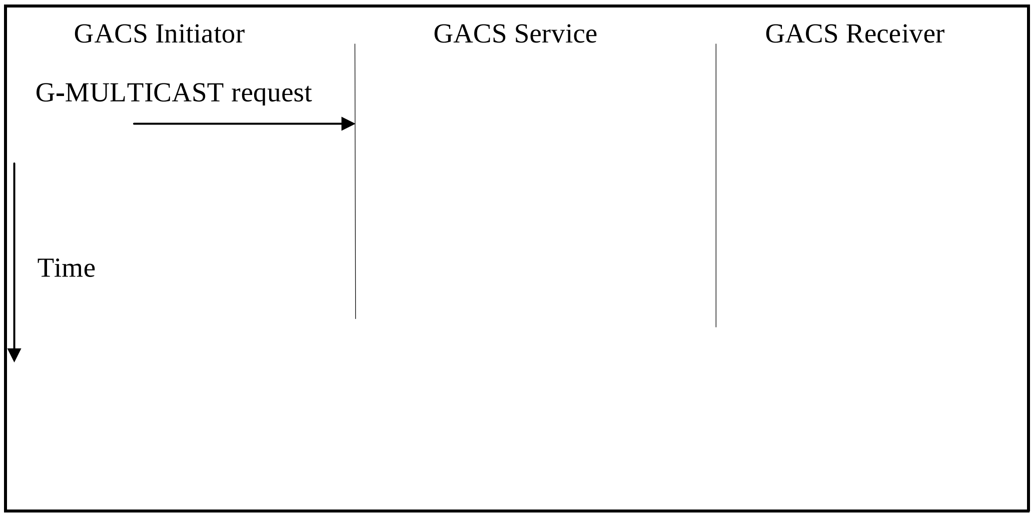


Figure 4.9.2-4. G-MULTICAST sequence diagram

Note 3.— The initiating GACS-User issues a G-MULTICAST request primitive at any time. The parameters of the G- MULTICAST primitives are specified in Table 4.9.2-5.

Table 4.9.2-5. G-MULTICAST parameters

Parameter Name	***Req***
Group Address	*M*
Toggle	*U*

Note 4.— The Group Address parameter specifies a PSAP address which includes in the supplied transport address a Group NSAP address.

Note 5.— The Toggle parameter is used to enable or disable the receipt of data addressed to the group address. The default is to enable such receipt. Valid abstract values are:

a) Enable Multicast Receipt,

b) Disable Multicast Receipt.

4.9.3 Protocol Definition

Note.— This section describes the format and sequencing of data allowed by the GACS protocol.

4.9.3.1 Model

Note 1.— The GACS service is provided by a protocol entity which is modelled as a finite state machine whose specification is given in this section. The GACS protocol entity communicates with its service-user by means of the GACS service primitives defined in section 4.9.2. It communicates with its Dialogue Service provider by means of the D- service primitives defined in 4.2 and 4.7.2.

Note 2.— The GACS protocol entity is driven by the receipt of input events from its GACS service-user, from its Dialogue Service provider that supports the communication relationships, and from an internal inactivity timer. The input events from the GACS service-user are GACS request primitives. The input events from the Dialogue Service provider are D-service indication and confirmation primitives which convey GACS Application Protocol Data Units (APDUs) from a peer GACS entity.

4.9.3.1.1 A new invocation of a GACS protocol entity shall be employed upon the receipt of a D-START indication primitive or a GACS primitive which requires a new dialogue to be established (i.e. any single-shot G-TRANSFER or G-TRANSFER-CONFIRMED request or a multi-shot G-TRANSFER or G-TRANSFER-CONFIRMED request where no suitable communications relationship currently exists).

4.9.3.2 Sequence Rules

4.9.3.2.1 The GACS protocol entity shall provide a service such that a user is able to invoke any GACS service request at any time when the protocol entity is operational.

4.9.3.2.2 The GACS protocol entity shall be capable of receiving and processing any valid GACS APDU at any time when the protocol entity is operational.

4.9.3.3 Timers

4.9.3.3.1 The GACS entity shall implement an inactivity timer T_inact for each communications relationship that is established using the "multi-shot" option of the G-TRANSFER service or G-TRANSFER-CONFIRMED service.

4.9.3.3.2 **Recommendation**.— *The threshold value of T_inact, referred to as T_inact_max, should be configurable.*

4.9.3.3.3 T_inact shall be reset to zero and started whenever a positive D-START confirmation primitive is received from the Dialogue Service Provider.

4.9.3.3.4 T_inact shall be reset to zero each time that user data is transferred to or from the Dialogue Service Provider.

4.9.3.3.5 If the value of T_inact reaches the threshold T_inact_max, then the GACS entity shall invoke the D-END service to end the relationship.

4.9.3.3.6 **Recommendation**.— *A configurable timer should be implemented so that dialogue establishment can be aborted by the GACS-Provider if no response is received after issuing a D-START request.*

4.9.3.3.7 **Recommendation**.— *A configurable timer should be implemented so that orderly dialogue termination can be aborted by the GACS-Provider if no response is received after issuing a D-END request.*

4.9.3.4 Elements of procedure

4.9.3.4.1 Handling of Level of Service

4.9.3.4.1.1 The GACS service shall map to either a connectionless provider or a connection-oriented provider with or without a logical acknowledgement, depending upon the value of the Requested Level of Service parameter, as shown in Table 4.9.3-1.

Table 4.9.3-1. Level of Service mappings

Requested Level of Service parameter	**Mapping to communications provider**
Single shot, no error recovery, unconfirmed service	Map to connectionless service, if available. User Data sent via D-UNIT-DATA service. Otherwise treat as "Single shot, error recovery, unconfirmed service."
Single shot, error recovery, unconfirmed service	Map to connection-oriented service. User Data sent via D-START service with negative D-START response.
Single shot, no error recovery, confirmed service	Map to connectionless service, if available. User Data and confirmation from GACS service sent via separate D-UNIT-DATA service invocations. Otherwise treat as "Single shot, error recovery, confirmed service."
Single shot, error recovery, confirmed service	Map to connection oriented service. User Data sent via D-START service with negative D-START response. Result passed to Initiator.
Multi shot, error recovery, unconfirmed service	Map to connection-oriented service. Existing connection used, or new connection established using D-START (see note). User Data sent via D-DATA service.
Multi-shot, error recovery, confirmed service	Map to connection oriented service. Existing connection used, or new connection established using D-START (see note). User Data sent via D-DATA service. Confirmation returned via D-DATA service invoked by GACS entity.

Note.— When using multi-shot mode, User Data is always sent using the D-DATA service. If necessary, a connection is first established independently of the User Data transfer. This allows the T-CONNECT optimisation to be realised, so that upper layer connections are established simultaneously with transport connections, and also allows a clean separation of User Data from connection set-up information. This can be beneficial when it is desired to negotiate a secure dialogue before sending any User Data.

4.9.3.4.2 Multiple Recipients

4.9.3.4.2.1 If more than one recipient is specified by the GACS-User in a service primitive invocation, this shall be treated as multiple sequential single-recipient invocations of the service.

4.9.3.4.3 Collisions

4.9.3.4.3.1 Collision of G-TRANSFER and/or G-TRANSFER-CONFIRMED services invoked by two peers shall result in the establishment of two independent communication links.

4.9.3.4.3.2 When two identical dialogues are established as a result of the above collision case, a G-END request issued for one of the dialogues shall be assumed to apply to both.

4.9.3.4.3.3 In the case of collisions between the G-TRANSFER and/or G-TRANSFER-CONFIRMED service on the one hand and the G-END service on the other hand invoked by two peers, the transfer shall be permitted to succeed, either over the existing link, or over a new communication link, depending upon the exact timing of the service invocations at the peer GACS entities.

Note.— Collision of the G-END service invoked by two peers will be internally resolved by the Dialogue service.

4.9.3.4.4 Disruptions

4.9.3.4.4.1 The services which are supported by connection-mode protocols may be disrupted by Abort events, either from the peer system or from the communication provider, and shall be able to handle such events.

4.9.3.4.4.2 If a GACS-User is waiting for confirmation of a previous data transfer, a provider abort event shall trigger such confirmation, with a status of confirmedUnknown, as it is not known whether the peer received the data before the abort.

4.9.3.4.5 Rules for extensibility

4.9.3.4.5.1 The protocol defined in this specification shall be designated Version 1 of the GACS protocol, and is identified as such by the absence of any extension fields in the abstract syntax.

4.9.3.4.5.2 When processing an incoming GACS APDU, the accepting protocol entity shall log the presence of any extension fields that are not defined in the abstract syntax of this protocol specification, and process the APDU as if the extensions were not present.

4.9.3.4.6 Exception handling

4.9.3.4.6.1 If a GACS PDU is received that cannot be decoded, then that PDU is considered an invalid PDU and exception handling procedures described in this section shall apply.

4.9.3.4.6.2 If a GACS PDU is received which is not an expected PDU at that time, then that PDU is considered an invalid PDU and exception handling procedures described in this section shall apply.

4.9.3.4.6.3 If delivery of a G-TRANSFER indication primitive is not possible due to local problems (e.g. resource limitations), then the receiving GACS entity shall discard the received data and invoke the exception handling procedures described in this section.

4.9.3.4.6.4 If delivery of a G-TRANSFER-CONFIRMED indication primitive is not possible due to local problems (e.g. resource limitations), then the receiving GACS entity shall discard the received data and return a confirmation indicating that the delivery failed, but otherwise continue as though no exception had occurred.

4.9.3.4.6.5 The exception handling shall result in the dialogue being aborted, if one exists, and a notification being given to the local GACS user, if possible.

4.9.3.5 GACS Protocol Data Unit Description

Note.— The GACS protocol uses a single PDU type to support all defined GACS services.

4.9.3.5.1 The fields of the GACS protocol data unit shall be used to support the GACS service as defined in this section.

4.9.3.5.2 The GACS protocol data unit shall consist of a header field and a User Data field.

4.9.3.5.3 The User Data field shall be used to convey the data passed to the GACS service via the User Data parameter transparently to the addressed recipient(s).

4.9.3.5.4 messageType

4.9.3.5.4.1 The messageType field of the GACS header shall be used to convey the Message Type parameter, passed to the GACS service by the user, transparently to the addressed recipient(s).

4.9.3.5.4.2 In confirmations generated by the GACS service provider, the messageType field shall have the same value as that in the GACS PDU whose receipt is being confirmed.

4.9.3.5.5 messageIdentifier

4.9.3.5.5.1 The messageIdentifier field of the GACS header shall be used to convey the Message Identifier parameter, passed to the GACS service by the user, transparently to the addressed recipient(s).

4.9.3.5.6 messageReference

4.9.3.5.6.1 The messageReference field of the GACS header shall be used to convey the Message Reference parameter, passed to the GACS service by the user, transparently to the addressed recipient(s).

4.9.3.5.6.2 In confirmations generated by the GACS service provider, the messageReference field shall have the same value as the messageIdentifier field in the GACS PDU whose receipt is being confirmed.

4.9.3.5.7 confirmation

4.9.3.5.7.1 The confirmation field of the GACS header shall be used to convey whether the GACS service user requested the G-TRANSFER service or the G-TRANSFER-CONFIRMED service.

4.9.3.5.7.2 In confirmation PDUs generated by the GACS service provider, the confirmation field shall be used to identify the PDU as a confirmation, and to state whether the confirmed PDU was passed to the receiving user without problems.

4.9.3.5.8 multiShot

4.9.3.5.8.1 The multiShot field of the GACS header shall be used to convey whether the GACS service user requested the "multi-shot" option of the CO G-TRANSFER service or G-TRANSFER-CONFIRMED service.

4.9.3.5.8.2 The multiShot field of the GACS header shall be also used to convey whether the GACS service user invoked the G-END service.

4.9.3.5.8.3 If a GACS user requests the multi-shot option, and the receiving (remote) GACS provider is able to support the option for this instance of communication, then a connection shall be maintained between the two GACS users until it is terminated either by one of the GACS users invoking the G-END service or by a T_inact timeout.

4.9.3.5.9 sender

4.9.3.5.9.1 The sender field of the GACS header shall be used to convey the identity of the sender when this has been requested by the user of the G-TRANSFER service or G-TRANSFER-CONFIRMED service, and in confirmations generated by the GACS service itself.

4.9.3.6 Formal Definition of GACS Protocol Data Units

Note.— This section specifies the protocol data units (PDUs) exchanged by GACS protocol entities, using Abstract Syntax Notation One (ASN.1). The notation is defined in ISO/IEC 8824-1 | ITU-T Rec. X.680 (1995).

4.9.3.6.1 The abstract syntax of the GACS protocol data units shall comply with the description contained in the ASN.1 module GACSProtocolVersion1, as defined in this section.

```
GACSProtocolVersion1 DEFINITIONS ::=
BEGIN
-- EXPORTS
-- everything
GACSpdu ::= SEQUENCE {
```

```
          gacsHeader              [0]       GACSHeaderType,
          userData                          [1]       OCTET STRING OPTIONAL
}
GACSHeaderType ::= SEQUENCE {
          messageType             [0]       GACSMessageId OPTIONAL,
          messageIdentifier       [1]       GACSMessageId OPTIONAL,
          messageReference        [2]       GACSMessageId OPTIONAL,
          confirmation            [3]       GACSConfirmation DEFAULT unconfirmedReq,
          multiShot                         [4]       GACSMultiShot DEFAULT single,
          sender                            [5]       GACSNameOrAddress OPTIONAL,
          ...
}
GACSMessageId ::= CHOICE {
          globalForm        [0] OBJECT IDENTIFIER,
          localForm         [1] INTEGER (0..255, ...),
...
}
GACSConfirmation ::= ENUMERATED {
          unconfirmedReq                    (0),
          confirmedReq                      (1),
          confirmedOk                       (2),
          confirmedNotOk                    (3),
          confirmedUnknown        (4),
          ...
}
GACSMultiShot ::= ENUMERATED {
          single            (0),
          maintained        (1),
          end               (2),
          ...
}
GACSNameOrAddress ::= CHOICE {
          name                              [1] ULCSPeerId,
          pSAPAddress             [2] OCTET STRING
}
ULCSPeerId ::= SEQUENCE {
          locationID        [1] ULCSLocationType,
          sysID             [2] INTEGER  OPTIONAL,
          -- sysID is the binary value of the concatenation of the bits of the LOC and SYS
          -- fields from the ATN NSAP address, with the LOC bits as the most significant.
          ...
```

```
}
ULCSLocationType ::= CHOICE {
        aircraft            [1] BIT STRING (SIZE(24)), -- 24-bit address
        groundFacility  [2] IA5String (SIZE(4..8)), -- ICAO designator
        ...
}
END -- of GACSProtocolVersion1
```

4.9.3.7 Encoding rules

4.9.3.7.1 The GACS protocol data units shall be encoded for transfer across the Dialogue Service boundary using the ASN.1 Packed Encoding Rules (PER) specified in ISO/IEC 8825-2 | ITU-T Rec. X.691 (1995), using the basic, unaligned variant.

Note.— Encoded GACS APDUs are treated as bit-oriented values that are not padded to an integral number of octets; the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type. The only exception to this, though not recommended, is that a GACS APDU may be treated as an octet-aligned single-ASN1-type when carried as user-information in an ACSE APDU.

4.9.3.8 GACS State Table

4.9.3.8.1 The GACS protocol entity shall behave as if it can exist only in one of the states defined in Table 4.9.3-2 for each instance of communication between two peer GACS-Users.

Table 4.9.3-2. States of the GACS Protocol Entity

State	**Short name**	**Description**
STA 0	Idle	No instance of communication; protocol machine is unallocated.
STA 1	Start-single	A D-START request has been submitted and a negative D-START confirmation is expected.
STA 2	Start-multi	A D-START request has been submitted and a positive D-START confirmation is expected.
STA 3	Ending	A D-END request has been submitted and a D-END confirmation is expected.
STA 4	Associated	A dialogue has been fully established with the peer

4.9.3.8.2 The GACS protocol entity shall be capable of detecting and processing each of the input events listed in Table 4.9.3-3.

Table 4.9.3-3. GACS Input Events

Event	**Description**
Unit Data req	G-TRANSFER request with Requested Level of Service set to (Single shot, no error recovery, unconfirmed service), or G-TRANSFER-CONFIRMED request with Requested Level of Service set to (Single shot, no error recovery, confirmed service) submitted by GACS-User
Single Shot req	G-TRANSFER request with Requested Level of Service set to (Single shot, error recovery, unconfirmed service), or G-TRANSFER-CONFIRMED request with Requested Level of Service set to (Single shot, error recovery, confirmed service) submitted by GACS-User
Multi-Shot req	G-TRANSFER request with Requested Level of Service set to (Multi shot, error recovery, unconfirmed service), or G-TRANSFER-CONFIRMED request with Requested Level of Service set to (Multi shot, error recovery, confirmed service) submitted by GACS-User
G-END req	G-END request submitted by GACS-User
D-START ind ()	D-START indication with no User Data delivered from Dialogue Service provider.
D-START ind (UD)	D-START indication, with User Data present delivered from Dialogue Service provider.
D-START cnf+	D-START confirmation from Dialogue Service provider, with Result = accepted
D-START cnf-	D-START confirmation with Result = rejected (transient) or rejected (permanent) delivered from Dialogue Service provider
D-DATA ind(cnf)	GACS PDU received via D-DATA indication with confirmation = confirmedOk delivered from Dialogue Service provider
D-DATA ind(UD)	GACS PDU received via D-DATA indication with confirmation = unconfirmedReq or confirmedReq delivered from Dialogue Service provider
D-UNIT-DATA ind(UD)	GACS PDU received via D-UNIT-DATA indication with confirmation = unconfirmedReq or confirmedReq delivered from Dialogue Service provider
D-UNIT-DATA ind(cnf)	GACS PDU received via D-UNIT-DATA indication with confirmation = confirmedOk delivered from Dialogue Service provider
D-END cnf+	D-END confirmation with Result = accepted delivered from Dialogue Service provider
D-END cnf-	D-END confirmation with Result = rejected delivered from Dialogue Service provider
D-END ind	D-END indication delivered from Dialogue Service provider
D-ABORT ind	D-ABORT indication delivered from Dialogue Service provider
D-P-ABORT ind	D-P-ABORT indication delivered from Dialogue Service provider
T-inact	Internal inactivity timer trigger

4.9.3.8.3 The GACS protocol entity shall exhibit external behaviour in accordance with the state table specified in Table 4.9.3-6, which shows diagrammatically the state transitions and actions performed by the protocol in response to incoming events, as follows:

a) Incoming events are shown in the first column of the state table, and are enumerated in Table 4.9.3-3. Valid protocol states are shown in the header row of the state table, and are described in Table 4.9.3-2.

b) When an input event occurs, the state transition and any action to be taken are indicated by the cell of the state table which is the intersection of the incoming event name and the current protocol state.

c) Each cell in the state table shows:

   1) the new state that the CF enters after the action has been performed, indicated by "STA I", where I is an integer

   2) optionally, one or more predicates, denoted "pN", where N is an integer. The state and action which follow the predicate are only valid if the predicate is TRUE. The inverse (logical NOT) of a predicate is indicated by the prefix "~" (tilde character).

   3) the action, if any, which is to be performed. The possible actions are described in Table 4.9.3-5.

d) Blank cells indicate error conditions.

e) Any of the input events which result in a new dialogue being established may occur at any time, and will result in a new instance of the state table being created, initially in the Idle state (STA 0).

4.9.3.8.4 For the purpose of this specification, the state table shall be treated as atomic, such that when an input event is invoked, that event is processed to completion within the same logical processing thread.

Note.— This provision does not imply any particular implementation architecture.

4.9.3.8.5 The following combinations of input events and states shall be treated as error conditions:

a) The occurrence of an input event other than those listed in Table 4.9.3-3; or

b) A combination of input event and current state which corresponds to a blank cell in Table 4.9.3-6.

4.9.3.8.6 In the event of a conflict between the actions implied by the state table and the text elsewhere in this specification, the text shall take precedence.

Table 4.9.3-4. Predicates

Predicate	Description
p1	Sender requested GACS-TRANSFER-CONFIRMED service (confirmation parameter = confirmedReq)
~p1	Sender requested GACS-TRANSFER (unconfirmed) service (confirmation parameter = unconfirmedReq)

Table 4.9.3-5. Output Actions

Action	Description
D-UNIT-DATAreq	Issue GACS PDU via D-UNIT-DATA request to Dialogue Service provider, with confirmation = unconfirmedReq or confirmedReq, as requested by GACS-User
D-UNIT-DATAreq(cnf)	Invoke D-UNIT-DATA request with GACS PDU as User Data to Dialogue Service provider, with confirmation = confirmedOk
D-STARTreq()	Invoke D-START request with no User Data to Dialogue Service provider
D-STARTreq (UD)	Invoke D-START request with GACS PDU as User Data to Dialogue Service provider, with confirmation = unconfirmedReq or confirmedReq, as requested by GACS-User
D-DATAreq(UD)	Invoke D-DATA request with GACS PDU as User Data to Dialogue Service provider, with confirmation = unconfirmedReq or confirmedReq, as requested by GACS-User
D-DATAreq(cnf)	Invoke D-DATA request with GACS PDU as User Data to Dialogue Service provider, with confirmation = confirmedOk
D-ABORTreq	Invoke D-ABORT request with no User Data to Dialogue Service provider
D-END req	Invoke D-END request with GACS PDU, if any, as User Data to Dialogue Service provider
D-STARTrsp+	Invoke D-START response with no User Data and Result = accepted to Dialogue Service provider
D-STARTrsp-(cnf)	Invoke D-START response with GACS PDU as User Data with confirmation = confirmedOk and Result = rejected (transient)
D-STARTrsp-	Invoke D-START response with no User Data and Result = rejected (transient) to Dialogue Service provider
D-ENDrsp+	Invoke D-END response with no User Data and Result = accepted to Dialogue Service provider
G-TFRind	Deliver G-TRANSFER indication to local GACS-User
G-TFR-CNFind	Deliver G-TRANSFER-CONFIRMED indication to local GACS-User
G-TFR-CNFcnf	Deliver G-TRANSFER-CONFIRMED confirmation to local GACS-User, with confirmation set to the value in the GACS PDU if present. If no PDU is present, set confirmation to confirmedNotOk. If P-P-ABORT was received, set confirmation to confirmedUnknown.
G-END ind	Deliver G-END indication to local GACS-User
t_inact(S)	Start the inactivity timer (from value zero)
t_inact(R)	Reset the inactivity timer to value zero
t_inact(O)	Stop the inactivity timer

Table 4.9.3-6. State Table

	STA 0 Idle	STA 1 Start-single	STA 2 Start-multi	STA 3 Ending	STA 4 Associated
Unit Data req	STA 0 D-UNIT-DATAreq				
Single Shot req	STA 1 D-STARTreq (UD)				
Multi-Shot req	STA 2 D-STARTreq()				STA 4 D-DATAreq (UD) t_inact(R)
G-END req	STA 0		STA 2	STA 3	STA 3 D-END req t_inact(R)
D-START ind ()	STA 4 D-STARTrsp+				
D-START ind (UD)	STA 0 ~p1: D-START rsp-, G-TFRind p1: D-STARTrsp-(cnf), G-TFR-CNFind				

D-START cnf+			STA 4
			D-DATAreq(UD)
			t_inact(S)		

D-START cnf-		STA 0 p1: G-TFR-CNFcnf	STA 0 p1: G-TFR-CNFcnf		
D-DATA ind(cnf)					STA 4 t_inact(R) p1:G-TFR-CNFcnf
D-DATA ind(UD)				STA 3 t_inact(R) ~p1:G-TFRind p1:G-TFR-CNFind	STA 4 t_inact(R) ~p1:G-TFRind p1:G-TFR-CNFind, D-DATAreq(cnf)
D-END cnf+				STA 0 t_inact(O)	
D-END cnf-					
D-END ind					STA 0 D-ENDrsp+ G-ENDind t_inact(O)
D-UNIT-DATA ind(cnf)	STA 0 G-TFR-CNFcnf				
D-UNIT-DATA ind(UD)	STA 0 ~p1:G-TFRind p1:G-TFR-CNFind, D-UNIT-DATAreq(cnf)				
D-ABORT ind		STA 0 p1: G-TFR-CNFcnf	STA 0 p1: G-TFR-CNFcnf	STA 0 t_inact(O)	STA 0 G-ENDind t_inact(O)
D-P-ABORT ind		STA 0 p1: G-TFR-CNFcnf	STA 0 p1: G-TFR-CNFcnf	STA 0 t_inact(O)	STA 0 G-ENDind t_inact(O)

	STA 0 Idle	STA 1 Start-single	STA 2 Start-multi	STA 3 Ending	STA 4 Associated
T-inact				STA 0 D-ABORTreq t_inact(O)	STA 3 D-END req t_inact(R)

4.9.3.9 GACS Protocol Description

4.9.3.9.1 On initiation, the GACS protocol entity shall be in the Idle state.

Note.— The following subsections describe the GACS protocol by describing the response to each of the possible input events in turn. Input events are: Request primitives from the GACS-User, Indication and Confirmation primitives from the Dialogue Service provider, and expiration of the inactivity timer.

4.9.3.9.2 G-TRANSFER Request primitive invoked by GACS-User

4.9.3.9.2.1 When invoked by the GACS-User, if the Recipient List parameter of the G-TRANSFER Request contains more than one element then the GACS protocol entity shall select each recipient in the list in turn and, for each recipient, behave as if the G-TRANSFER Request primitive had been invoked with only that element present in the Recipient List parameter.

4.9.3.9.2.2 If a G-TRANSFER request primitive is invoked and the Requested Level of Service parameter indicates "multi-shot" mode, and a compatible dialogue already exists with the identified peer entity, then the primitive shall be processed by the protocol entity which is associated with the existing dialogue and is in the "Associated" state (STA 4).

Note.— A "compatible" dialogue is one whose QoS parameters (Class of Communication, Priority and RER), GACS-User Version Number and Security Requirements parameter match the corresponding parameters requested in the G-TRANSFER request primitive.

4.9.3.9.2.3 Apart from the case identified in the preceding paragraph (for re-use of an existing dialogue), the G-TRANSFER request primitive shall be processed by a new invocation of the protocol entity, which by definition is in the "Idle" state (STA 0).

4.9.3.9.2.4 If a G-TRANSFER request primitive is invoked with the Requested Level of Service parameter containing the abstract value "Single shot, no error recovery, unconfirmed service," and the connectionless dialogue service is available, and the GACS entity is in the Idle state (STA 0), then the GACS entity shall:

a) Create a GACSpdu APDU with field values defined in Table 4.9.3-7.

b) At the supporting Dialogue Service boundary, invoke a D-UNIT-DATA request primitive with parameters as defined in Table 4.9.3-8.

c) Remain in the Idle state.

Table 4.9.3-7

GACSpdu field	**Value**
messageType	Message Type from the G-TRANSFER request primitive, if provided. Absent otherwise.
messageIdentifier	Message Identifier from the G-TRANSFER request primitive, if provided. Absent otherwise.
messageReference	Message Reference from the G-TRANSFER request primitive, if provided. Absent otherwise.
confirmation	Absent (default abstract value “unconfirmedReq”)
multiShot	Absent (default abstract value “single”)
sender	Sender identity if requested in the G-TRANSFER request. Absent otherwise.
userData	User Data from the G-TRANSFER request primitive, if provided. Absent otherwise.

Table 4.9.3-8

D-UNIT-DATA / D-START request parameter	**Status**	**Value**
Called Peer ID Called Sys-ID Called Presentation Address	U C U	One element of the Recipient List parameter value from the G-TRANSFER request. One and only one of Peer ID and Presentation Address is given. Sys-ID is present if Peer ID is, and requested by the user.
Calling Peer ID Calling Sys-ID Calling Presentation Address	U C U	Not used.
DS-User Version Number	U	GACS-User Version Number from the G-TRANSFER request, if provided. Absent otherwise.
Security Requirements	U	Security Requirements from the G-TRANSFER request, if provided. Absent otherwise.
QOS: Routing Class	M	*Class of Communication* from the G-TRANSFER request.
QOS: Priority	M	*Priority* from the G-TRANSFER request.
QOS: Residual Error Rate	C	*RER* from the G-TRANSFER request, if provided. The default is to request the highest available integrity. (Only significant if the Class of Communication parameter has a non-ATSC value).
User Data	M / U	The GACSpdu APDU created according to Table 4.9.3-7.

4.9.3.9.2.5 If a G-TRANSFER request primitive is invoked with the Requested Level of Service parameter containing the abstract value “Single shot, no error recovery, unconfirmed service” and the connectionless dialogue service is not available, or with the Requested Level of Service parameter containing the abstract value “Single shot, error recovery, unconfirmed service”, and the GACS entity is in the Idle state (STA 0), then the GACS entity shall:

a) Create a GACSpdu APDU with field values defined in Table 4.9.3-7.

b) At the supporting Dialogue Service boundary, invoke a D-START request primitive with parameters as defined in Table 4.9.3-8.

c) Enter the Start-single state (STA 1).

4.9.3.9.2.6 If a G-TRANSFER request primitive is invoked with the Requested Level of Service parameter containing the abstract value “Multi shot, error recovery, unconfirmed service”, and the GACS entity is in the Idle state (STA 0), then the GACS entity shall:

a) Create a GACSpdu APDU with field values defined in Table 4.9.3-9 and store it until a dialogue with the specified peer has been successfully established.

b) At the supporting Dialogue Service boundary, invoke a D-START request primitive with parameters as defined in Table 4.9.3-10.

c) Enter the Start-multi state (STA 2).

Table 4.9.3-9

GACSpdu field	**Value**
messageType	Message Type from the G-TRANSFER request primitive, if provided. Absent otherwise.
messageIdentifier	Message Identifier from the G-TRANSFER request primitive, if provided. Absent otherwise.
messageReference	Message Reference from the G-TRANSFER request primitive, if provided. Absent otherwise.
confirmation	Absent (default abstract value “unconfirmedReq”)
multiShot	"maintained"
sender	Sender identity if requested in the G-TRANSFER request. Absent otherwise.
userData	User Data from the G-TRANSFER request primitive, if provided. Absent otherwise.

Table 4.9.3-10

D-START request parameter	**Status**	**Value**
Called Peer ID Called Sys-ID Called Presentation Address	U C U	One element of the Recipient List parameter value from the G-TRANSFER request. One and only one of Peer ID and Presentation Address is given. Sys-ID is present if Peer ID is, and requested by the user.
Calling Peer ID Calling Sys-ID Calling Presentation Address	U C U	Not used.
DS-User Version Number	U	GACS-User Version Number from the G-TRANSFER request, if provided. Absent otherwise.
Security Requirements	U	Security Requirements from the G-TRANSFER request, if provided. Absent otherwise.
QOS: Routing Class	M	*Class of Communication* from the G-TRANSFER request.
QOS: Priority	M	*Priority* from the G-TRANSFER request.
QOS: Residual Error Rate	C	*RER* from the G-TRANSFER request, if provided.. The default is to request the highest available integrity. (Only significant if the Class of Communication parameter has a non-ATSC value).
User Data	U	Absent

4.9.3.9.2.7 If a G-TRANSFER request primitive is invoked with the Requested Level of Service parameter containing the abstract value "Multi shot, error recovery, unconfirmed service", and the GACS entity is in the Associated state (STA 4), then the GACS entity shall:

a) Reset timer t_inact,

b) Create a GACSpdu APDU with field values defined in Table 4.9.3-9,

c) At the supporting Dialogue Service boundary, invoke a D-DATA request with the GACSpdu APDU as the D-DATA *User Data* parameter value, and

d) Remain in the Associated state (STA 4).

4.9.3.9.3 G-TRANSFER-CONFIRMED Request primitive invoked by GACS-User

4.9.3.9.3.1 When invoked by the GACS-User, if the Recipient List parameter of the G-TRANSFER-CONFIRMED Request contains more than one element then the GACS protocol entity shall select each recipient in the list in turn and, for each recipient, behave as if the G-TRANSFER-CONFIRMED Request primitive had been invoked with only that element present in the Recipient List parameter.

4.9.3.9.3.2 If a G-TRANSFER-CONFIRMED request primitive is invoked and the Requested Level of Service parameter indicates "multi-shot" mode, and a compatible dialogue already exists with the identified peer entity, then the primitive shall be processed by the protocol entity which is associated with the existing dialogue and is in the "Associated" state (STA 4).

Note.— A "compatible" dialogue is one whose QoS parameters (Class of Communication, Priority and RER), GACS-User Version Number and Security Requirements parameter match the corresponding parameters requested in the G-TRANSFER-CONFIRMED request primitive.

4.9.3.9.3.3 Apart from the case identified in the preceding paragraph (for re-use of an existing dialogue), the G-TRANSFER-CONFIRMED request primitive shall be processed by a new invocation of the protocol entity, which by definition is in the "Idle" state (STA 0).

4.9.3.9.3.4 If a G-TRANSFER-CONFIRMED request primitive is invoked with the Requested Level of Service parameter containing the abstract value "Single shot, no error recovery, confirmed service" and the connectionless dialogue service is available, and the GACS entity is in the Idle state (STA 0), then the GACS entity shall:

a) Create a GACSpdu APDU with field values defined in Table 4.9.3-11.

b) At the supporting Dialogue Service boundary, invoke a D-UNIT-DATA request primitive with parameters as defined in Table 4.9.3-12.

c) Remain in the Idle state.

Table 4.9.3-11

GACSpdu field	**Value**
messageType	Message Type from the G-TRANSFER-CONFIRMED request primitive, if provided. Absent otherwise.
messageIdentifier	Message Identifier from the G-TRANSFER-CONFIRMED request primitive.
messageReference	Message Reference from the G-TRANSFER-CONFIRMED request primitive, if provided. Absent otherwise.
confirmation	"confirmedReq"
multiShot	Absent (default abstract value "single")
sender	Sender identity if requested in the G-TRANSFER-CONFIRMED request. Absent otherwise.
userData	User Data from the G-TRANSFER-CONFIRMED request primitive, if provided. Absent otherwise.

Table 4.9.3-12

D-UNIT-DATA / D-START request parameter	**Status**	**Value**
Called Peer ID Called Sys-ID Called Presentation Address	U C U	One element of the Recipient List parameter value from the G-TRANSFER-CONFIRMED request. One and only one of Peer ID and Presentation Address is given. Sys-ID is present if Peer ID is, and requested by the user.
Calling Peer ID Calling Sys-ID Calling Presentation Address	U C U	Not used.
DS-User Version Number	U	GACS-User Version Number from the G-TRANSFER-CONFIRMED request, if provided. Absent otherwise.
Security Requirements	U	Security Requirements from the G-TRANSFER-CONFIRMED request, if provided. Absent otherwise.
QOS: Routing Class	M	*Class of Communication* from the G-TRANSFER-CONFIRMED request.
QOS: Priority	M	*Priority* from the G-TRANSFER-CONFIRMED request.
QOS: Residual Error Rate	C	*RER* from the G-TRANSFER-CONFIRMED request, if provided. The default is to request the highest available integrity. (Only significant if the Class of Communication parameter has a non-ATSC value).
User Data	M / U	The GACSpdu APDU created according to Table 4.9.3-11.

4.9.3.9.3.5 If a G-TRANSFER-CONFIRMED request primitive is invoked with the Requested Level of Service parameter containing the abstract value “Single shot, no error recovery, confirmed service” and the connectionless dialogue service is not available, or with the Requested Level of Service parameter containing the abstract value “Single shot, error recovery, confirmed service”, and the GACS entity is in the Idle state (STA 0), then the GACS entity shall:

a) Create a GACSpdu APDU with field values defined in Table 4.9.3-11

b) At the supporting Dialogue Service boundary, invoke a D-START request primitive with parameters as defined in Table 4.9.3-12.

c) Enter the Start-single state (STA 1).

4.9.3.9.3.6 If a G-TRANSFER-CONFIRMED request primitive is invoked with the Requested Level of Service parameter containing the abstract value “Multi shot, error recovery, confirmed service”, and the GACS entity is in the Idle state (STA 0), then the GACS entity shall:

a) Create a GACSpdu APDU with field values defined in Table 4.9.3-13 and store it until a dialogue with the specified peer has been successfully established.

b) At the supporting Dialogue Service boundary, invoke a D-START request primitive with parameters as defined in Table 4.9.3-14.

c) Enter the Start-multi state (STA 2).

Table 4.9.3-13

GACSpdu field	**Value**
messageType	Message Type from the G-TRANSFER-CONFIRMED request primitive, if provided. Absent otherwise.
messageIdentifier	Message Identifier from the G-TRANSFER-CONFIRMED request primitive.
messageReference	Message Reference from the G-TRANSFER-CONFIRMED request primitive, if provided. Absent otherwise.
confirmation	“confirmedReq”
multiShot	“maintained”
sender	Sender identity if requested in the G-TRANSFER-CONFIRMED request. Absent otherwise.
userData	User Data from the G-TRANSFER-CONFIRMED request primitive, if provided. Absent otherwise.

Table 4.9.3-14

D-START request parameter	**Status**	**Value**
Called Peer ID Called Sys-ID Called Presentation Address	U C U	One element of the Recipient List parameter value from the G-TRANSFER-CONFIRMED request. One and only one of Peer ID and Presentation Address is given. Sys-ID is present if Peer ID is, and requested by the user.
Calling Peer ID Calling Sys-ID Calling Presentation Address	U C U	Not used.
DS-User Version Number	U	GACS-User Version Number from the G-TRANSFER-CONFIRMED request, if provided. Absent otherwise.

D-START request parameter	**Status**	**Value**
Security Requirements	U	Security Requirements from the G-TRANSFER-CONFIRMED request, if provided. Absent otherwise.
QOS: Routing Class	M	*Class of Communication* from the G-TRANSFER-CONFIRMED request.
QOS: Priority	M	*Priority* from the G-TRANSFER-CONFIRMED request.
QOS: Residual Error Rate	C	*RER* from the G-TRANSFER-CONFIRMED request, if provided. The default is to request the highest available integrity. (Only significant if the Class of Communication parameter has a non-ATSC value).
User Data	U	Absent

4.9.3.9.3.7 If a G-TRANSFER-CONFIRMED request primitive is invoked with the Requested Level of Service parameter containing the abstract value "Multi shot, error recovery, confirmed service", and the GACS entity is in the Associated state (STA 4), then the GACS entity shall:

a) Reset timer t_inact,

b) Create a GACSpdu APDU with field values defined in Table 4.9.3-13.

c) At the supporting Dialogue Service boundary, invoke a D-DATA request with the GACSpdu APDU as the D-DATA User Data parameter value, and

d) Remain in the Associated state (STA 4).

4.9.3.9.4 G-END Request primitive invoked by GACS-User

4.9.3.9.4.1 When invoked by the GACS-User, if the Recipient List parameter of the G-END Request contains more than one element then the GACS protocol entity shall select each recipient in the list in turn and, for each recipient, behave as if the G-END Request primitive had been invoked with only that element present in the Recipient List parameter.

4.9.3.9.4.2 If a G-END request primitive is invoked and the GACS entity for the Dialogue which corresponds to the Recipient List entry is in the Associated state (STA 4), then the GACS entity shall:

a) Reset timer t_inact to value zero,

b) Create a GACSpdu APDU with field values defined in Table 4.9.3-15,

c) At the supporting Dialogue Service boundary, invoke a D-END request for the dialogue which corresponds to the Recipient List entry, with the GACSpdu APDU as the D-END *User Data* parameter value, and

d) Enter the Ending state (STA 3).

Table 4.9.3-15

GACSpdu field	Value
messageType	Message Type from the G-END request primitive, if provided. Absent otherwise.
messageIdentifier	Absent.
messageReference	Message Reference from the G-END request primitive, if provided. Absent otherwise.
confirmation	Absent (default value "unconfirmedReq").
multiShot	"end"
sender	The local location identity (syntax: ULCSLocationType) value if available. Absent otherwise.
userData	Absent.

4.9.3.9.4.3 If a G-END request primitive is invoked and the GACS entity for the Dialogue which corresponds to the Recipient List entry is in the Start-multi state (STA 2), then the GACS entity shall:

a) Remain in the same state, and

b) If possible, indicate to the G-END invoker by local means (e.g. procedure return code) that the requested operation was not performed, due to a pending transfer.

4.9.3.9.4.4 If a G-END request primitive is invoked and the GACS entity for the Dialogue which corresponds to the Recipient List entry is in the Idle state (STA 0) or the Ending state (STA 3), then the GACS entity shall take no action and remain in the same state.

4.9.3.9.5 D-UNIT-DATA Indication primitive invoked by supporting service

4.9.3.9.5.1 Upon receipt of a D-UNIT-DATA indication primitive, if the GACS entity is in the Idle state (STA 0), and the APDU contained in the D-UNIT-DATA User Data parameter is a valid GACSpdu APDU with the confirmation APDU-element set to the abstract value "unconfirmedReq," and the multishot APDU-element set to the abstract value "single", then the GACS entity shall:

a) Deliver a G-TRANSFER indication to the local GACS-User, with parameters as defined in Table 4.9.3-16.

b) Remain in the Idle state (STA 0).

Table 4.9.3-16

G-TRANSFER indication parameter	Status	Value
Sender	C(=)	The GACSpdu sender parameter value, if provided. Absent otherwise.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Identifier	C(=)	The GACSpdu messageIdentifier parameter value, if provided. Absent otherwise.
Message Reference	C(=)	The GACSpdu messageReference parameter value, if provided. Absent otherwise.
GACS-User Version Number	C(=)	DS-User Version Number from the D-UNIT-DATA indication, if provided. Absent otherwise.
Security Requirements	C(=)	Security Requirements from the D-UNIT-DATA indication, if provided. Absent otherwise.
Class of Communication	M(=)	The Routing Class field from the Quality of Service parameter of the D-UNIT-DATA indication.
Priority	M(=)	The Priority field from the Quality of Service parameter of the D-UNIT-DATA indication.
RER	C(=)	The Residual Error Rate field from the Quality of Service parameter of the D-UNIT-DATA indication.
Requested Level of Service	M(=)	"Single shot, no error recovery, unconfirmed service"
User Data	C(=)	userData field from the GACSpdu if provided. Absent otherwise.

4.9.3.9.5.2 Upon receipt of a D-UNIT-DATA indication primitive, if the GACS entity is in the Idle state (STA 0), and the APDU contained in the D-UNIT-DATA User Data parameter is a valid GACSpdu APDU with the confirmation APDU-element set to the abstract value "confirmedReq," and the multishot APDU-element set to the abstract value "single", then the GACS entity shall:

a) Deliver a G-TRANSFER-CONFIRMED indication to the local GACS-User, with parameters as defined in Table 4.9.3-17.

b) Create a confirmation GACSpdu APDU with field values defined in Table 4.9.3-18,

c) At the supporting Dialogue Service boundary, invoke a D-UNIT-DATA request primitive with parameters as defined in Table 4.9.3-19.

d) Remain in the Idle state (STA 0).

Table 4.9.3-17

G-TRANSFER-CONFIRMED indication parameter	**Status**	**Value**
Sender	C(=)	The GACSpdu sender parameter value, if provided. Absent otherwise.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Identifier	M(=)	The GACSpdu messageIdentifier parameter value.
Message Reference	C(=)	The GACSpdu messageReference parameter value, if provided. Absent otherwise.
GACS-User Version Number	C(=)	DS-User Version Number from the D-UNIT-DATA indication, if provided. Absent otherwise.
Security Requirements	C(=)	Security Requirements from the D-UNIT-DATA indication, if provided. Absent otherwise.
Class of Communication	M(=)	The Routing Class field from the Quality of Service parameter of the D-UNIT-DATA indication.
Priority	M(=)	The Priority field from the Quality of Service parameter of the D-UNIT-DATA indication.
RER	C(=)	The Residual Error Rate field from the Quality of Service parameter of the D-UNIT-DATA indication.
Requested Level of Service	M(=)	"Single shot, no error recovery, confirmed service"
User Data	C(=)	userData field from the GACSpdu if provided. Absent otherwise.

Table 4.9.3-18

GACSpdu field	**Value**
messageType	The messageType parameter value in the received GACSpdu, if provided. Absent otherwise.
messageIdentifier	Absent.
messageReference	Message Identifier from the received GACSpdu.
confirmation	If the G-TRANSFER-CONFIRMED indication is capable of being delivered then "confirmedOk", otherwise "confirmedNotOk"

GACSpdu field	**Value**
multiShot	Absent. (Default value "single").
sender	The local location identity (syntax: ULCSLocationType) value if available. Absent otherwise.
userData	Absent.

Table 4.9.3-19

D-UNIT-DATA request parameter	**Status**	**Value**
Called Peer ID Called Sys-ID Called Presentation Address	U C U	Set equal to the Calling Peer ID, Calling Sys-ID and Calling Presentation Address parameters of the received D-UNIT-DATA indication.
Calling Peer ID Calling Sys-ID Calling Presentation Address	U C U	Not used.
DS-User Version Number	U	The local GACS-User Version Number, if available. Absent otherwise.
Security Requirements	U	As for the Security Requirements in the G-TRANSFER-CONFIRMED indication (Table 4.9.3-17).
QOS: Routing Class	M	As for the Class of Communication in the G-TRANSFER-CONFIRMED indication (Table 4.9.3-17).
QOS: Priority	M	As for Priority in the G-TRANSFER-CONFIRMED indication (Table 4.9.3-17).
QOS: Residual Error Rate	M	As for RER in the G-TRANSFER-CONFIRMED indication (Table 4.9.3-17).
User Data	M	The GACSpdu APDU created according to Table 4.9.3-18.

4.9.3.9.5.3 Upon receipt of a D-UNIT-DATA indication primitive, if the GACS entity is in the Idle state (STA 0), and the APDU contained in the D-UNIT-DATA User Data parameter is a valid GACSpdu APDU with the confirmation APDU-element set to one of the abstract values ("confirmedOk", "confirmedNotOk", or "confirmedUnknown") and no userData APDU-element present, then the GACS entity shall:

a) Deliver a G-TRANSFER-CONFIRMED confirmation to the local GACS-User, with parameters as defined in Table 4.9.3-20, and

b) Remain in the Idle state (STA 0).

Table 4.9.3-20

G-TRANSFER-CONFIRMED confirmation parameter	**Status**	**Value**
Sender	M	The GACSpdu sender parameter value, if provided. Otherwise, the Presentation Address of the peer.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Reference	M	The GACSpdu messageReference parameter value.
GACS-User Version Number	C	The DS-User Version Number from the D-UNIT-DATA indication, if provided. Absent otherwise.
Result	M	"Successful delivery", "Delivery failed", or "Service problem - delivery status uncertain", for values of the GACSpdu confirmation parameter of "confirmedOk", "confirmedNotOk", or "confirmedUnknown", respectively.

4.9.3.9.6 D-START Indication primitive delivered by supporting service

4.9.3.9.6.1 Upon receipt of a D-START indication primitive, if the GACS entity is in the Idle state (STA 0), and there is no D-START User Data parameter present, then the GACS entity shall:

a) At the supporting Dialogue Service boundary, invoke a D-START response primitive with parameters as defined in Table 4.9.3-21, and

b) If the Result parameter is "accepted" then enter the Associated state (STA 4), otherwise remain in the Idle state (STA 0).

Table 4.9.3-21

D-START response parameter	**Status**	**Value**
DS-User Version Number	U	The local GACS-User Version Number, if available. Absent otherwise.
Security Requirements	U	The local Security Requirements, if available. Absent otherwise.
QOS: Routing Class	U	*Routing Class* from the Quality of Service parameter of the received D-START indication
QOS: Priority	U	*Priority* from the Quality of Service parameter of the received D-START indication
QOS: Residual Error Rate	U	*Residual Error Rate* from the Quality of Service parameter of the received D-START indication.

D-START response parameter	**Status**	**Value**
Result	M	If the local GACS entity is able to accept the dialogue, then "accepted". If, for local reasons, the GACS entity is unable to accept the dialogue, then "rejected (transient)" or "rejected (permanent)", depending upon severity.
User Data	U	Absent

4.9.3.9.6.2 Upon receipt of a D-START indication primitive, if the GACS entity is in the Idle state (STA 0), and the APDU contained in the D-START User Data parameter is a valid GACSpdu APDU with the confirmation APDU-element set to the abstract value "unconfirmedReq," then the GACS entity shall:

a) Deliver a G-TRANSFER indication to the local GACS-User, with parameters as defined in Table 4.9.3-22,

b) At the supporting Dialogue Service boundary, invoke a D-START response primitive with parameters as defined in Table 4.9.3-23, and

c) Remain in the Idle state (STA 0).

Table 4.9.3-22

G-TRANSFER indication parameter	**Status**	**Value**
Sender	C(=)	The GACSpdu sender parameter value, if provided. Absent otherwise.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Identifier	C(=)	The GACSpdu messageIdentifier parameter value, if provided. Absent otherwise.
Message Reference	C(=)	The GACSpdu messageReference parameter value, if provided. Absent otherwise.
GACS-User Version Number	C(=)	DS-User Version Number from the received D-START indication, if provided. Absent otherwise.
Security Requirements	C(=)	Security Requirements from the received D-START indication, if provided. Absent otherwise.
Class of Communication	M(=)	The Routing Class field from the Quality of Service parameter of the received D-START indication.
Priority	M(=)	The Priority field from the Quality of Service parameter of the received D-START indication.

G-TRANSFER indication parameter	**Status**	**Value**
RER	C(=)	The Residual Error Rate field from the Quality of Service parameter of the received D-START indication.
Requested Level of Service	M(=)	"single shot, error recovery, unconfirmed service"
User Data	C(=)	userData field from the GACSpdu if provided. Absent otherwise.

Table 4.9.3-23

D-START response parameter	**Status**	**Value**
DS-User Version Number	U	The local GACS-User Version Number, if available. Absent otherwise.
Security Requirements	U	The local Security Requirements, if available. Absent otherwise.
QOS: Routing Class	U	*Routing Class* from the Quality of Service parameter of the received D-START indication
QOS: Priority	U	*Priority* from the Quality of Service parameter of the received D-START indication
QOS: Residual Error Rate	U	*Residual Error Rate* from the Quality of Service parameter of the received D-START indication.
Result	M	"rejected (transient)"
User Data	U	Absent

4.9.3.9.6.3 Upon receipt of a D-START indication primitive, if the GACS entity is in the Idle state (STA 0), and the APDU contained in the D-START User Data parameter is a valid GACSpdu APDU with the confirmation APDU-element set to the abstract value "confirmedReq," then the GACS entity shall:

a) Deliver a G-TRANSFER-CONFIRMED indication to the local GACS-User, with parameters as defined in Table 4.9.3-24,

b) Create a confirmation GACSpdu APDU with field values defined in Table 4.9.3-25,

c) At the supporting Dialogue Service boundary, invoke a D-START response primitive with parameters as defined in Table 4.9.3-26, and

d) Remain in the Idle state (STA 0).

Table 4.9.3-24

G-TRANSFER-CONFIRMED indication parameter	**Status**	**Value**
Sender	C(=)	The GACSpdu sender parameter value, if provided. Absent otherwise.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Identifier	M(=)	The GACSpdu messageIdentifier parameter value.
Message Reference	C(=)	The GACSpdu messageReference parameter value, if provided. Absent otherwise.
GACS-User Version Number	C(=)	DS-User Version Number from the received D-START indication, if provided. Absent otherwise.
Security Requirements	C(=)	Security Requirements from the received D-START indication, if provided. Absent otherwise.
Class of Communication	M(=)	The *Routing Class* field from the Quality of Service parameter of the received D-START indication.
Priority	M(=)	The *Priority* field from the Quality of Service parameter of the received D-START indication.
RER	C(=)	The *Residual Error Rate* field from the Quality of Service parameter of the received D-START indication.
Requested Level of Service	M(=)	"single shot, error recovery, confirmed service"
User Data	C(=)	userData field from the GACSpdu if provided. Absent otherwise.

Table 4.9.3-25

GACSpdu field	**Value**
messageType	As for the Message Type in the G-TRANSFER-CONFIRMED indication (Table 4.9.3-24).
messageIdentifier	Absent.
messageReference	Message Identifier from the received GACSpdu.
confirmation	If the G-TRANSFER-CONFIRMED indication is capable of being delivered then "confirmedOk", otherwise "confirmedNotOk".
multiShot	Absent. (Default value "single").
sender	The local location identity (syntax: ULCSLocationType) value if available. Absent otherwise.
userData	Absent.

Table 4.9.3-26

D-START response parameter	**Status**	**Value**
DS-User Version Number	U	The local GACS-User Version Number, if available. Absent otherwise.
Security Requirements	U	The local Security Requirements, if available. Absent otherwise.
QOS: Routing Class	U	Routing Class from the Quality of Service parameter of the received D-START indication
QOS: Priority	U	Priority from the Quality of Service parameter of the received D-START indication
QOS: Residual Error Rate	U	Residual Error Rate from the Quality of Service parameter of the received D-START indication.
Result	M	"rejected (transient)"
User Data	U	The GACSpdu APDU (Table 4.9.3-25)

4.9.3.9.7 D-START Confirmation primitive delivered by supporting service

4.9.3.9.7.1 Upon receipt of a D-START Confirmation primitive, if the GACS entity is in the Start-multi state (STA 2), and the D-START Result parameter has the abstract value "accepted," and the D-START User Data parameter is absent, then the GACS entity shall:

a) At the supporting Dialogue Service boundary, invoke a D-DATA request primitive with the previously-formed and stored GACSpdu APDU (see 4.9.3.9.2.6 or 4.9.3.9.3.6) as the User Data parameter.

b) Start the T_inact inactivity timer from value zero,

c) Enter the Associated state (STA 4).

4.9.3.9.7.2 Upon receipt of a D-START Confirmation primitive, if the GACS entity is in the Start-single state (STA 1) or the Start-multi state (STA 2), and the D-START Result parameter has the abstract value "rejected (transient)" or "rejected (permanent)," then the GACS entity shall:

a) If the local GACS-User is waiting for confirmation of a previous G-TRANSFER-CONFIRMED request, and the User Data parameter of the received D-START Confirmation contains a valid GACSpdu APDU, then deliver a G-TRANSFER-CONFIRMED confirmation to the local GACS-User, with parameters as defined in Table 4.9.3-27.

b) If the local GACS-User is waiting for confirmation of a previous G-TRANSFER-CONFIRMED request, and the User Data parameter of the received D-START Confirmation does not contain a valid GACSpdu APDU, then deliver a G-TRANSFER-CONFIRMED confirmation to the local GACS-User, with parameters as defined in Table 4.9.3-28.

c) Enter the Idle state (STA 0).

Table 4.9.3-27

G-TRANSFER-CONFIRMED confirmation parameter	**Status**	**Value**
Sender	M	The GACSpdu sender parameter value, if provided. Otherwise, the Presentation Address of the peer.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Reference	M	The GACSpdu messageReference parameter value.
GACS-User Version Number	C	DS-User Version Number from the received D-START confirmation, if provided. Absent otherwise.
Result	M	"Successful delivery" if the confirmation field of the received GACSpdu has value "confirmedOk". "Delivery failed" otherwise.

Table 4.9.3-28

G-TRANSFER-CONFIRMED confirmation parameter	**Status**	**Value**
Sender	M	The location corresponding to the Called Peer, in the same format (Presentation Address or ULCSPeerID) used in the original D-START request.
Message Type	C(=)	Absent
Message Reference	M	A local default value.
GACS-User Version Number	C	DS-User Version Number from the received D-START confirmation, if provided. Absent otherwise.
Result	M	"Service problem - delivery status uncertain"

4.9.3.9.8 D-DATA Indication primitive delivered by supporting service

4.9.3.9.8.1 Upon receipt of a D-DATA Indication primitive, if the GACS entity is in the Associated state (STA 4), and the D-DATA User Data parameter contains a valid GACSpdu APDU with the confirmation

APDU-element set one of the values ("confirmedOk", "confirmedNotOk" or "confirmedUnknown") then the GACS entity shall:

a) Reset the T_inact inactivity timer to value zero,

b) If the local GACS-User had previously issued a G-TRANSFER-CONFIRMED request, and the messageReference field of the received GACS APDU matches the Message Identifier parameter of that request, then deliver a G-TRANSFER-CONFIRMED confirmation to the local GACS-User, with parameters as defined in Table 4.9.3-29.

c) Remain in the Associated state (STA 4).

Table 4.9.3-29

G-TRANSFER-CONFIRMED confirmation parameter	**Status**	**Value**
Sender	M	The GACSpdu sender parameter value, if provided. Otherwise, the Presentation Address of the peer.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Reference	M	The GACSpdu messageReference parameter value.
GACS-User Version Number	C	The peer DS-User Version Number obtained when the dialogue was established, if known. Absent otherwise.
Result	M	"Successful delivery" if the confirmation field of the received GACSpdu has value "confirmedOk". "Delivery failed" otherwise.

4.9.3.9.8.2 Upon receipt of a D-DATA Indication primitive, if the GACS entity is in the Associated state (STA 4) or the Ending state (STA 3), and the D-DATA User Data parameter contains a valid GACSpdu APDU with the confirmation APDU-element set to the abstract value "unconfirmedReq," then the GACS entity shall:

a) Reset the T_inact inactivity timer to value zero,

b) Deliver a G-TRANSFER indication to the local GACS-User, with parameters as defined in Table 4.9.3-30, and

c) Remain in the same state.

Table 4.9.3-30

G-TRANSFER indication parameter	**Status**	**Value**
Sender	C(=)	The GACSpdu sender parameter value, if provided. Absent otherwise.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Identifier	C(=)	The GACSpdu messageIdentifier parameter value, if provided. Absent otherwise.
Message Reference	C(=)	The GACSpdu messageReference parameter value, if provided. Absent otherwise.
GACS-User Version Number	C(=)	If provided when the dialogue was established, the peer DS-User Version Number. Absent otherwise.
Security Requirements	C(=)	If provided when the dialogue was established, the peer Security Requirements. Absent otherwise.
Class of Communication	M(=)	The Routing Class field from the Quality of Service parameter when the dialogue was established.
Priority	M(=)	The Priority field from the Quality of Service parameter when the dialogue was established.
RER	C(=)	The Residual Error Rate field from the Quality of Service parameter when the dialogue was established.
Requested Level of Service	M(=)	"multi shot, error recovery, unconfirmed service"
User Data	C(=)	userData field from the GACSpdu if provided. Absent otherwise.

4.9.3.9.8.3 Upon receipt of a D-DATA Indication primitive, if the GACS entity is in the Associated state (STA 4) or the Ending state (STA 3), and the D-DATA User Data parameter contains a valid GACSpdu APDU with the confirmation APDU-element set to the abstract value "confirmedReq," then the GACS entity shall:

a) Reset the T_inact inactivity timer to value zero,

b) Deliver a G-TRANSFER-CONFIRMED indication to the local GACS-User, with parameters as defined in Table 4.9.3-31, and

c) If in the Associated state (STA 4):

   1) create a confirmation GACSpdu APDU with field values defined in Table 4.9.3-18, and

   2) at the supporting Dialogue Service boundary, invoke a D-DATA request primitive with the newly created GACSpdu APDU as the User Data parameter.

d) Remain in the same state.

Table 4.9.3-31

G-TRANSFER-CONFIRMED indication parameter	**Status**	**Value**
Sender	C(=)	The GACSpdu sender parameter value, if provided. Absent otherwise.
Message Type	C(=)	The GACSpdu messageType parameter value, if provided. Absent otherwise.
Message Identifier	M(=)	The GACSpdu messageIdentifier parameter value.
Message Reference	C(=)	The GACSpdu messageReference parameter value, if provided. Absent otherwise.
GACS-User Version Number	C(=)	If provided when the dialogue was established, the peer DS-User Version Number. Absent otherwise.
Security Requirements	C(=)	If provided when the dialogue was established, the peer Security Requirements. Absent otherwise.
Class of Communication	M(=)	The Routing Class field from the Quality of Service parameter when the dialogue was established.
Priority	M(=)	The Priority field from the Quality of Service parameter when the dialogue was established.
RER	C(=)	The Residual Error Rate field from the Quality of Service parameter when the dialogue was established.
Requested Level of Service	M(=)	"multi shot, error recovery, confirmed service"
User Data	C(=)	userData field from the GACSpdu if provided. Absent otherwise.

4.9.3.9.9 D-END Indication primitive delivered by supporting service

4.9.3.9.9.1 Upon receipt of a D-END Indication primitive, if the GACS entity is in the Associated state (STA 4), then the GACS entity shall:

a) Stop the T_inact inactivity timer,

b) Deliver a G-END indication to the local GACS-User, with parameters as defined in Table 4.9.3-32,

c) At the supporting Dialogue Service boundary, invoke a D-END response primitive with the Result parameter set to "accepted" and no User Data parameter, and

d) Enter the Idle state (STA 0).

Table 4.9.3-32

G-END indication parameter	**Status**	**Value**
Sender	M	If the D-END User Data contains a GACSpdu, the value of the GACSpdu *sender* parameter, if present. Otherwise the assumed sender based on local context information. Syntax: ULCSLocationType if available, Presentation Address otherwise.
Message Type	C(=)	If the D-END User Data contains a GACSpdu, the value of the GACSpdu *messageType* parameter, if present. Absent otherwise.
Message Reference	C(=)	If the D-END User Data contains a GACSpdu, the value of the GACSpdu *messageReference* parameter, if present. Absent otherwise.

4.9.3.9.10 D-END Confirmation primitive delivered by supporting service

4.9.3.9.10.1 Upon receipt of a D-END Confirmation primitive with the result parameter having value "accepted", if the GACS entity is in the Ending state (STA 3), then the GACS entity shall:

a) Stop the T_inact inactivity timer,

b) Enter the Idle state (STA 0).

4.9.3.9.11 D-ABORT Indication or D-P-ABORT primitive delivered by supporting service

4.9.3.9.11.1 Upon receipt of a D-ABORT Indication or D-P-ABORT Indication primitive, for a dialogue which is not in the Idle state (STA 0), the GACS entity shall:

a) Stop the T_inact inactivity timer if it is running,

b) If in state Start-single (STA 1) or Start-multi (STA 2), and there is an outstanding G-TRANSFER-CONFIRMED confirmation, then deliver a G-TRANSFER-CONFIRMED confirmation to the local GACS-User, with parameters as defined in Table 4.9.3-33.

c) If in state Associated (STA 4), then deliver a G-END indication to the local GACS-User, with parameters as defined in Table 4.9.3-34,

d) Enter the Idle state (STA 0).

Table 4.9.3-33

G-TRANSFER-CONFIRMED confirmation parameter	**Status**	**Value**
Sender	M	The location of the peer GACS entity based on local context information. Syntax: ULCSLocationType if available, Presentation Address otherwise.
Message Type	C(=)	Absent.
Message Reference	M	Local value indicating abnormal termination.
GACS-User Version Number	C	Absent.
Result	M	"Service problem - delivery status uncertain"

Table 4.9.3-34

G-END indication parameter	**Status**	**Value**
Sender	M	The location of the peer GACS entity based on local context information. Syntax: ULCSLocationType if available, Presentation Address otherwise.
Message Type	C(=)	Absent.
Message Reference	C(=)	Absent.

4.9.3.9.12 Timer T_inact expiry

4.9.3.9.12.1 If the inactivity timer T_inact expires for a given dialogue, the GACS entity shall:

a) If in state Associated (STA 4), then

1) Reset T_inact to value zero, while keeping it running,

2) Create a GACSpdu APDU with field values defined in Table 4.9.3-35,

3) At the supporting Dialogue Service boundary, invoke a D-END request with the GACSpdu APDU as the D-END *User Data* parameter value, and

4) Enter the Ending state (STA 3).

b) If in state Ending (STA 3), then

1) Stop T_inact,

2) At the supporting Dialogue Service boundary, invoke a D-ABORT request primitive with the originator parameter set to "provider" and no User Data parameter.

3) Enter the Idle state (STA 0).

Table 4.9.3-35

GACSpdu field	**Value**
messageType	Absent.
messageIdentifier	Absent.
messageReference	Absent.
confirmation	Absent (default value "unconfirmedReq").
multiShot	"end"
sender	The local location identity (syntax: ULCSLocationType) value if available. Absent otherwise.
userData	Absent.

4.9.4 Communication Requirements

Note 1.— This section contains the requirements that the GACS ASE application imposes on the underlying communication system.

Note 2.— The GACS specification makes use of the Dialogue Service (DS) as defined in 4.2 and 4.7.2. This is the abstract service which ATN Application ASEs use to interact with the UL communications service. That is, the DS is the combination of specific internal primitives made available by the Control Function (CF) at the lower boundary of the ATN ASE/ASO - it is the application's "world view". In order to provide this service, the CF uses the services of ACSE.

4.9.4.1 **Recommendation.**— *Where both connection-oriented and connectionless communications providers are available, implementations of GACS and associated Upper Layers should be capable of being configured to support either the same TSAP address or different TSAP addresses for CO and CL transport services, respectively.*

4.9.4.2 Use of Connectionless Provider

4.9.4.2.1 When a connectionless communications provider is available and selected, GACS service primitives shall map onto D-UNIT-DATA service primitives.

Note.— There are constraints on the connectionless service data length. The Transport CL user data is limited to 63,488 octets per TSDU, and this limit will be reflected in the GACS service. If a GACSpdu APDU exceeds the size constraints of the D-UNIT-DATA service, then it will be necessary to use a connection-oriented communications provider, if available. Otherwise, it is the responsibility of the GACS-User to ensure that size constraints are respected.

4.9.4.3 Use of Connection-Oriented Provider

4.9.4.3.1 When a connection-oriented communications provider is available and selected, GACS service primitives shall map onto the connection-oriented Dialogue Service primitives (D-START, D-DATA, D-END, D-ABORT and D-P-ABORT).

4.9.4.4 Mapping to Dialogue Service Parameters

4.9.4.4.1 The GACS parameter Class of Communication shall map to the D-START or D-UNIT-DATA Routing Class QoS parameter using the values specified in the "Security Tag Value" column of Table 5.6-1.

4.9.4.4.2 The GACS parameter Priority shall map to the D-START or D-UNIT-DATA Priority QoS parameter using the values specified in the "Transport Layer Priority" column of Table 1-2.

4.9.4.4.3 Each list member of the GACS parameter Recipient List shall map to a separate invocation of the equivalent D-START or D-UNIT-DATA Called Peer ID, Called Sys-ID and Called Presentation Address parameters, depending upon the recipient format used by the GACS-User.

4.9.4.4.4 The Calling Peer ID, Calling Sys-ID and Calling Presentation Address parameters of the D-START and D-UNIT-DATA primitives shall be unused.

4.9.4.4.5 The GACS parameter User Version Number shall map to the D-START or D-UNIT-DATA DS-User Version Number parameter.

4.9.4.4.6 If no GACS-User Version Number is specified, then the DS-User Version Number shall be left unused in D-START and D-UNIT-DATA primitives.

4.9.4.4.7 The GACS parameter Security Requirements shall map to the D-START or D-UNIT-DATA Security Requirements parameter.

4.9.4.4.8 The GACS parameter RER shall map to the D-START or D-UNIT-DATA RER QoS parameter.

4.9.4.4.9 If no RER value is specified, then the DS RER QoS parameter shall be set to the logical value that maps to the lowest RER (highest integrity) supported by the transport service.

4.9.4.4.10 The GACS PDU, as defined in section 4.9.3.5 and 4.9.3.6 shall map to the D-DATA, D-START or D-UNIT-DATA User Data parameter.

4.9.5 User Requirements

Note.— This section outlines the requirements that a user of a GACS ASE must meet.

4.9.5.1 **Recommendation**.— *A configurable timer should be implemented so that the GACS-User can take appropriate action if no confirmation is received after issuing a G-TRANSFER-CONFIRMED request.*

4.9.5.2 **Recommendation**.— *The GACS-User should set the Message Type parameter in GACS primitives to a value which unambiguously identifies the abstract syntax used by that GACS User.*

Note.— The above recommendation facilitates the sharing of the GACS service by more than one GACS-User.

4.9.6 Subsetting Rules

Note.— This section specifies conformance requirements which all implementations of the GACS protocol obey.

4.9.6.1 An implementation of the GACS service claiming conformance to the specified protocol shall be realised either as a GACS-AE or a GACS-ASO as shown in Table 4.9.6-1.

Table 4.9.6-1. Conformant realisations

Conformant Configuration Ref.	**Functionality Description**
GACS-AE	GACS is a complete addressable ATN application accessible by user applications via an exposed interface.
GACS-ASO	GACS is an element of the ATN upper layers which is not directly addressable. It provides an enhanced dialogue service to user ASEs.

4.9.6.2 An implementation of the GACS service claiming conformance to the specified protocol shall support the GACS protocol features as shown in Table 4.9.6-2.

Table 4.9.6-2. Conformant communication subsets

Conformant Configuration Ref.	**Functionality Description**
I	Both connection-oriented and connectionless underlying stacks supported.
II	Only connection-oriented underlying stack supported.
III	Only connectionless underlying stack supported.

Note 1.— The GACS service user is not made aware which of the subsets is implemented.

Note 2.— Configurations I and III give more efficient handling of the "single shot, no error recovery" level of service.

Note 3.— Implementations of Configuration I will not be able to interwork with implementations of Configuration II in cases where the "single shot, no error recovery" level of service is invoked by the GACS service user. Implementations of Configuration II will always be able to interwork with implementations of Configuration I.

Note 4.— Implementations of Configuration III will only be able to support the "single shot, no error recovery" level of service, and will not be able to interwork with implementations of Configuration II. Implementations of Configuration III will always be able to interwork with implementations of Configuration I.