The ATN SARPs

Sub-Volume II

Air-Ground Applications

Editors' Draft

Please note that this is the final editor's draft circulated within the ATNP. This text will be passed to ICAO for publication. However, it should be noted that this text in no way replaces the ICAO version nor can it be considered of equal status. The official definitive version is that published in hardcopy by ICAO and all claims of compliance must be made against that version.

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

Please check our Web Site regularly for information on updates to the draft SARPs

Errata and Disclaimer

Please note that this document was prepared from a number of separate files prepared by different editors. The initial page numbers in these files are not completely synchronised, and we have made no attempt to change this, in order to avoid problems with references from Working Papers. You may therefore find some overlap between pages numbers, when those pages came from different files.

The preparation of this document has been 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, noted that actual implementation and operational experience was yet to be gained by the international civil aviation community. In this regard, the ANC agreed that the detailed part of ATN SARPs should be published as an ICAO manual, while retaining its SARPs-style language. The ANC will review the status of the document, in its entirety or in parts, after sufficient implementation and operational experience has been gained and the requirements for further standardization, in the interests of safety, regularity and efficiency of international civil aviation have been better ascertained.

This document consists of five 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)

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

TABLE OF CONTENTS

SUB-VOLUME II. AIR-GROUND APPLICATIONS

2.1 CONTEXT MANAGEMENT APPLICATION

2.1.1 INTRODUCTION

The CM application allows addressing capability for data link applications. The CM application provides the capability to establish a logon between peer ATS ground systems and ATS ground and aircraft systems. Once an appropriate connection is established, CM provides data link application information, the capability to log-on to another ground system, and the capability to update log-on information.

Note 1.— Structure

a) 2.1.1: INTRODUCTION contains the purpose, structure, and a summary of the functions of CM.

b) 2.1.2: GENERAL REQUIREMENTS contains CM ASE Version Number and error processing requirements.

c) 2.1.3: ABSTRACT SERVICE DEFINITION contains the description of the abstract service provided by the CM Application Service Element (CM-ASE).

d) 2.1.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by CM-ASEs using Abstract Syntax Notation Number One (ASN.1).

e) 2.1.5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the CM protocol, as well as time constraints and CM-ASE protocol descriptions. State tables are also included.

f) 2.1.6: COMMUNICATION REQUIREMENTS contains the requirements that the CM application imposes on the underlying communication system.

g) 2.1.7: CM USER REQUIREMENTS contains requirements imposed on the user of the CM-ASE service.

h) 2.1.8: SUBSETTING RULES contains the conformance requirements which all implementations of the CM protocol obey.

Note 2.— Functional Descriptions

a) Logon Functional Description

1) The Logon function can only be air initiated. The aircraft system can use the logon function to provide an application name and version number for each air-only initiated application, and an application name, address, and version number for each application that the aircraft wishes to use that can be ground initiated, along with flight plan information as required by the ground system. In response, the ground provides an application name for each ground-only initiated requested application and an application name,

address and version number for each requested application that can be air initiated and that the ground can support.

2) Up to a maximum of 256 applications can be supported.

3) Each time a logon is accomplished between a given aircraft and a ground system, the latest exchanged information replaces any previous information for each indicated application.

4) The CM Logon Request message provides required flight plan information, the aircraft's CM application name and address, and information for each application for which data link services are desired. For each application that can be ground initiated the aircraft must provide the application name, version number and address. For each application which is only air initiated the aircraft must provide the application name and version number.

5) The CM Logon Response message provides information for the logon-indicated air-initiated applications. For each desired air-initiated application the ground provides the application name, version number, and address.

b) Update Functional Description

1) This function provides a method for the ground system to update application information. This function assumes that the logon function has been accomplished.

2) The CM Update message can provide updated ground information for up to 256 applications. For each updated application the ground provides the application's name, version number and address.

c) Contact Functional Description

1) This function provides a method for the ground system to request the aircraft system to initiate the logon function with a designated ground system. It is expected that the contact function will only be used when ground connectivity is not available between respective ground system applications. This function assumes that the logon function has been accomplished with the ground system initiating the contact function. The ground initiates this function with a contact request specifying which ground system to logon with. The aircraft initiates a logon as specified above and indicates the success or lack thereof of the logon.

2) The CM Contact Request message provides the ground system CM application address that the initiating ground system is requesting the aircraft to logon with.

3) The CM Contact Response message provides the information indicating whether or not the requested contact was successful.

d) Forwarding Functional Description

1) This function provides a method for a ground system to forward aircraft information received from the CM Logon function to another ground system. This function is initiated by a ground system, which supports ground-ground forwarding, having completed a successful logon that can then forward the aircraft CM Logon information to other ground systems. It is a one-way forwarding of information with an indication of success, failure or non-support from the receiving ground system. If the ground system receiving this CM information supports ground-ground forwarding, it can then initiate a CM Update function to provide information to the aircraft for any air-initiated applications.

2) The CM Forward Request message contains the information as provided in the initial logon.

e) Registration Functional Description

1) This function provides a method for the air and ground CM applications to make available the application name, address, and version number for each application exchanged in the logon, update or forward functions to other applications or communications systems in the aircraft or on the ground.

2) There are no message exchanges for this function.

2.1.2 GENERAL REQUIREMENTS

2.1.2.1 CM ASE Version Number

2.1.2.1.1 The CM-air-ASE and CM-ground-ASE version numbers shall both be set to one.

2.1.2.2 Error Processing Requirements

2.1.2.2.1 In the event of information input by the CM-user being incompatible with that able to be processed by the system, the CM-user shall be notified.

2.1.2.2.2 In the event of a CM-user invoking a CM service primitive when the CM-ASE is not in a state specified in 2.1.5, the CM-user shall be notified.

2.1.3 THE ABSTRACT SERVICE

2.1.3.1 Service Description

2.1.3.1.1 An implementation of either the CM ground based service or the CM air based service shall exhibit external behaviour consistent with having implemented a CM-ground-ASE or CM-air-ASE respectively.

Note 1.— 2.1.3 defines the abstract service interface for the CM service. The CM-ASE abstract service is described from the viewpoint of the CM-air-user, the CM-ground-user and the CM-service-provider.

Note 2.— 2.1.3 defines the static behaviour (i.e., the format) of the CM abstract service. Its dynamic behaviour (i.e., how it is used) is described in 2.1.7.

Note 3.— Figure 2.1.3-1 shows the functional model of the CM Application. The functional modules identified in this model are the following:

a) the CM-user,

b) the CM Application Entity (CM-AE) service interface,

c) the CM-AE,

d) the CM Control Function (CM-CF),

e) the CM Application Service Element (CM-ASE) service interface,

f) the CM-ASE, and

g) the Dialogue Service (DS) interface.

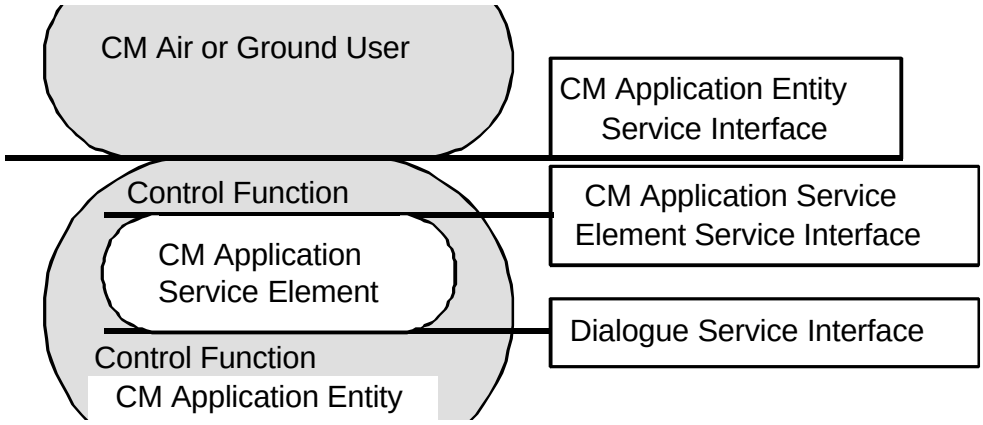

Figure 2.1.3-1. Functional Model of the CM Application

Note 4.— The CM-user represents the operational part of the CM system. This user does not perform the communication functions but relies on a communication service provided to it via the CM-AE through the CM-AE service interface. The individual actions at this interface are called CM-AE service primitives. Similarly, individual actions at other interfaces in the communication system are called service primitives at these interfaces.

Note 5.— The CM-AE consists of several elements including the CM-ASE and the CM-CF. The DS interface is made available by the CM-CF to the CM-ASE for communication with the peer CM-ASE.

Note 6.— The CM-ASE is the element in the communication system which executes the CM specific protocol. In other words, it takes care of the CM specific service primitive sequencing actions, message creation, timer management, error and exception handling.

Note 7.— The CM-ASE interfaces only with the CM-CF. This CM-CF is responsible for mapping service primitives received from one element (such as the CM-ASE and the CM-user) to other elements which interface with it. The part of the CM-CF which is relevant from the point of view of the CM application, i.e. the part between the CM-user and the CM-ASE, will map CM-AE service primitives to CM-ASE service primitives transparently.

Note 8.— The DS interface is the interface between the CM-ASE and the part of CM-CF underneath the CM-ASE, and provides the dialogue service as defined in 4.2.

2.1.3.2 The CM-ASE Abstract Service

Note.— There is no requirement to implement the service in a CM product; however, it is necessary to implement the ground based and air based system in such a way that it will be impossible to detect (from the peer system) whether or not an interface has been built.

2.1.3.2.1 The CM-ASE abstract service shall consist of a set of the following services as allowed by the subsetting rules in 2.1.8:

a) CM-logon service as defined in 2.1.3.3,

b) CM-update service as defined in 2.1.3.4,

c) CM-contact service as defined in 2.1.3.5,

d) CM-end service as defined in 2.1.3.6,

e) CM-forward service as defined in 2.1.3.7,

f) CM-user-abort service as defined in 2.1.3.8, and

g) CM-provider-abort service as defined in 2.1.3.9.

Note 1.— For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables in 2.1.3.

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 left being present, and equal to that value;*

d) ***M*** *mandatory;*

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

f) ***U*** *user option.*

Note 2.— The following abbreviations are used:

a) ***Req*** *- request; data is input by CM-user initiating the service to its respective ASE,*

b) ***Ind*** *- indication; data is indicated by the receiving ASE to its respective CM-user,*

c) ***Rsp*** *- response; data is input by receiving CM-user to its respective ASE, and*

d) ***Cnf*** *- confirmation; data is confirmed by the initiating ASE to its respective CM-user.*

Note 3.— An unconfirmed service allows a message to be transmitted in one direction, without providing a corresponding response.

Note 4.— A confirmed service provides end-to-end confirmation that a message sent by one user was received by its peer user.

Note 5.— An abstract syntax is a syntactical description of a parameter which does not imply a specific implementation. Only when the CM-ASE maps a parameter onto an APDU field, or vice versa, is the abstract syntax of the parameter described by using the ASN.1 of 2.1.4 for this field.

2.1.3.3 CM-logon Service

Note.— The CM-logon service allows the CM-air-user to initiate data link service. The CM-air-user provides information on each data link application for which it desires a data link service. The CM-ground-user responds indicating whether or not the CM-logon was successful, and if successful, includes information on each data link application it can support. It is a confirmed service.

2.1.3.3.1 If the CM-air-ASE version number is less than or equal to the CM-ground-ASE, then the CM-logon service shall contain the primitives and parameters as presented in Table 2.1.3-1.

Table 2.1.3-1. CM-logon Service Parameters Air-ASE version Number ≤ Ground-ASE Version Number

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Facility Designation	M			
Aircraft Address	M	M(=)		
CM ASE Version Number		C		
Logon Request	M	M(=)		
Logon Response			M	M(=)
Class of Communication Service	U			
Maintain Dialogue			U	C(=)

2.1.3.3.2 If the CM-air-ASE version number is greater than the CM-ground-ASE, then the CM-logon service shall contain the primitives and parameters as presented in Table 2.1.3-2.

Table 2.1.3-2. CM-logon Service Parameters Air-ASE version Number > Ground-ASE Version Number

Parameter Name	**Req**	**Cnf**
Facility Designation	M	
Aircraft Address	M	
CM ASE Version Number		M
Logon Request	M	
Class of Communication Service	U	

2.1.3.3.3 Facility Designation

Note.— This parameter contains the addressed ground system's facility designation.

2.1.3.3.3.1 The *Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.1.3.3.4 Aircraft Address

Note.— This parameter contains 24-bit aircraft address of the aircraft initiating the CM-logon service.

2.1.3.3.4.1 The *Aircraft Address* parameter value shall conform to the abstract syntax 24-bit aircraft address.

2.1.3.3.5 CM ASE Version Number

Note.— This parameter contains the version number of the CM-ASE.

2.1.3.3.5.1 When provided by the CM-ASE, the *Version Number* parameter shall conform to an abstract integer value from 1 to 255.

2.1.3.3.5.2 Only if the CM-air-ASE version number is less than the CM-ground-ASE version number shall the CM-air-ASE version number be indicated to the CM-ground-user.

2.1.3.3.5.3 Only if the CM-air-ASE version number is greater than the CM-ground-ASE version number shall the CM-ground-ASE version number be confirmed to the CM-air-user.

Note 1.— If the CM-air-ASE version number is the same as the CM-ground-ASE version number, the Version Number *parameter is not present in the indication given to the CM-ground-user, nor in the confirmation to the CM-air-user.*

Note 2.— The CM-air-ASE and CM-ground-ASE version numbers are both set to 1.

2.1.3.3.6 Logon Request

Note.— The Logon Request *parameter contains the following data:*

a) information for each data link application available on the aircraft, for which the aircraft requires data link service, and

b) aircraft flight plan information (e.g. flight id, aircraft destination and departure airport and time) as required by the addressed ground system.

2.1.3.3.6.1 The *Logon Request* parameter value shall conform to the ASN.1 abstract syntax CMLogonRequest.

2.1.3.3.7 Logon Response

Note.— This parameter contains information for each requested data link application for which the ground is able to provide data link service.

2.1.3.3.7.1 The *Logon Response* parameter value shall conform to the ASN.1 abstract syntax CMLogonResponse.

2.1.3.3.8 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service if specified by the CM-air-user.

2.1.3.3.8.1 When this parameter is specified by the CM-air-user, the *Class of Communication Service* parameter value shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G”, or “H”.

Note.— If not specified by the CM-air-user, this indicates that there is no routing preference.

2.1.3.3.9 Maintain Dialogue

Note 1.— This parameter is used to indicate whether or not the requested CM dialogue is to remain open after a Logon Response.

Note 2.— Whenever a CM dialogue is kept open by the CM-ground-user, it must later be explicitly closed by the CM-ground-user.

Note 3.— This parameter is only provided by the CM-ground-user when the CM-ground-user wishes to keep the CM dialogue open.

2.1.3.3.9.1 If provided by the CM-ground-user this parameter shall have the abstract value “maintain”.

2.1.3.4 CM-update Service

Note.— The CM-update service allows the CM-ground-user to transmit updated ground information for its applications to update previously coordinated CM-logon information. It is an unconfirmed service.

2.1.3.4.1 The CM-update service shall contain the primitives and parameters as presented Table 2.1.3-3.

Table 2.1.3-3. CM-update Service Parameters

Parameter Name	**Req**	**Ind**
Aircraft Address	C	
Facility Designation	C	C(=)
Update Information	M	M(=)
Class of Communication Service	U	

2.1.3.4.2 Aircraft Address

Note.— This parameter contains the addressed aircraft's 24-bit aircraft address.

2.1.3.4.2.1 The *Aircraft Address* parameter value shall conform to the abstract syntax 24-bit aircraft address.

2.1.3.4.2.2 If a CM dialogue does not exist when a CM-ground user invokes the CM-update service request, the CM-ground-user shall provide the *Aircraft Address* parameter value.

Note.— The CM-update service does not use this parameter when a CM dialogue exists.

2.1.3.4.3 Facility Designation

Note.— This parameter contains the ground system's facility designation.

2.1.3.4.3.1 The *Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.1.3.4.3.2 If a CM dialogue does not exist when a CM-ground user invokes the CM-update service request, the CM-ground-user shall provide the *Facility Designation* parameter value.

Note.— The CM-update service does not use this parameter when a CM dialogue exists.

2.1.3.4.4 Update Information

Note.— This parameter contains information on each updated data link application.

2.1.3.4.4.1 The *Update Information* parameter value shall conform to the ASN.1 abstract syntax CMUpdate.

2.1.3.4.5 Class of Communication Service

Note 1.— This parameter contains the value of the required class of communication service if specified by the CM-ground-user.

Note 2.— The CM-update service does not use this parameter when a CM dialogue exists.

2.1.3.4.5.1 Where specified by the CM-ground-user, the *Class of Communication Service* parameter shall have one of the following abstract values: "A", "B", "C", "D", "E", "F", "G", or "H".

Note.— If not specified by the CM-ground-user, this indicates that there is no routing preference.

2.1.3.5 CM-contact Service

Note.— The CM-contact service allows the CM-ground-user, after successful completion of a CM logon, to request that an aircraft logon with another ground system. It is a confirmed service.

2.1.3.5.1 The CM-contact service shall contain the primitives and parameters as presented in Table 2.1.3-4.

Table 2.1.3-4. CM-contact Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Aircraft Address	C			
Facility Designation	C	C(=)		
Contact Request	M	M(=)		
Contact Response			M	M(=)
Class of Communication Service	U			

2.1.3.5.2 Aircraft Address

Note.— This parameter contains the addressed aircraft's 24-bit aircraft address.

2.1.3.5.2.1 The *Aircraft Address* parameter value shall conform to the abstract syntax 24-bit aircraft address.

2.1.3.5.2.2 If a CM dialogue does not exist when a CM-ground user invokes the CM-contact service request, the CM-ground-user shall provide the *Aircraft Address* parameter value.

Note.— The CM-contact service does not use this parameter when a CM dialogue exists.

2.1.3.5.3 Facility Designation

Note.— This parameter contains the ground system's facility designation.

2.1.3.5.3.1 The *Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.1.3.5.3.2 If a CM dialogue does not exist when a CM-ground user invokes the CM-contact service request, the CM-ground-user shall provide the *Facility Designation* parameter value.

Note.— The CM-contact service does not use this parameter when a CM dialogue exists.

2.1.3.5.4 Contact Request

Note.— This parameter contains the facility designation for the ground system that the CM-ground-user requests the aircraft to contact.

2.1.3.5.4.1 The *Contact Request* parameter value shall conform to the ASN.1 abstract syntax CMContactRequest.

2.1.3.5.5 Contact Response

Note.— This parameter indicates success, or lack thereof, of the requested contact.

2.1.3.5.5.1 The *Contact Response* parameter value shall conform to the ASN.1 abstract syntax CMContactResponse.

2.1.3.5.6 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service if specified by the CM-ground-user.

2.1.3.5.6.1 When this parameter is specified by the CM-ground-user, the *Class of Communication Service* parameter value shall have one of the following abstract values: "A", "B", "C", "D", "E", "F", "G", or "H".

Note.— If not specified by the CM-ground-user, this indicates that there is no routing preference.

2.1.3.6 CM-end Service

Note 1.— This service provides the capability for the CM-ground-user to terminate a CM dialogue. This service is only needed when the CM-ground-user maintains a CM dialogue during the logon process. It is an unconfirmed service.

Note 2.— Only the CM-ground-user will be capable of initiating the CM-end service.

2.1.3.6.1 The CM-end service shall contain the primitives as presented Table 2.1.3-5.

Table 2.1.3-5. CM-end Service Parameters

Parameter Name	Req	Ind
none		

2.1.3.7 CM-forward Service

Note.— The CM-forward service allows a CM-ground-user to forward data received in a CM-logon request to another CM-ground system. It is a confirmed service.

2.1.3.7.1 If the sending CM-ground-ASE version number is less than or equal to the receiving CM-ground-ASE version number, then the CM-forward service shall contain the primitives and parameters as presented in Table 2.1.3-6.

Table 2.1.3-6. CM-forward Service Parameters Sending Ground-ASE Version Number ≤ Receiving Ground-ASE Version Number

Parameter Name	Req	Ind	Cnf
Called Facility Designation	M		
Calling Facility Designation	M	M(=)	
CM ASE Version Number		C	
Forward Request	M	M(=)	
Class of Communication Service	U		
Result			M

2.1.3.7.2 If the sending CM-ground-ASE version number is greater than the receiving CM-ground-ASE version number, then the CM-forward service shall contain the primitives and parameters as presented in Table 2.1.3-7.

Table 2.1.3-7. CM-forward Service Parameters Sending Ground-ASE Version Number > Receiving Ground-ASE Version Number

Parameter Name	Req	Cnf
Called Facility Designation	M	
Calling Facility Designation	M	
CM ASE Version Number		M
Forward Request	M	
Class of Communication Service	U	
Result		M

2.1.3.7.3 Called Facility Designation

Note.— This parameter contains the receiving ground system's facility designation.

2.1.3.7.3.1 The *Called Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.1.3.7.4 Calling Facility Designation

Note.— This parameter contains the sending ground system's facility designation.

2.1.3.7.4.1 The *Calling Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.1.3.7.5 CM ASE Version Number

Note.— This parameter contains the version number of the CM-ground-ASE.

2.1.3.7.5.1 When provided by the CM-ASE, the *Version Number* parameter shall conform to an abstract integer value from 1 to 255.

2.1.3.7.5.2 Only if the sending CM-ground-ASE version number is less than the receiving CM-ground-ASE version number shall the sending CM-ground-ASE version number be indicated to the receiving CM-ground-user.

2.1.3.7.5.3 Only if the sending CM-ground-ASE version number is greater than the receiving CM-ground-ASE version number shall the receiving CM-ground-ASE version number be confirmed to the sending CM-ground-user.

Note 1.— If the sending CM-ground-ASE version number is the same as the receiving CM-ground-ASE version number, the Version Number *parameter is not present in the indication given to the receiving CM-ground-user, nor in the confirmation to the sending CM-ground-user.*

Note 2.— The sending CM-ground-ASE and receiving CM-ground-ASE version numbers are both set to 1.

2.1.3.7.6 Forward Request

Note.— This parameter contains information as provided in the CM Logon Request.

2.1.3.7.6.1 The *Forward Request* parameter value shall conform to the ASN.1 abstract syntax CMForwardRequest.

2.1.3.7.7 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service if specified by the initiating CM-ground-user.

2.1.3.7.7.1 When this parameter is specified by the CM-ground-user, the *Class of Communication Service* parameter value shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G”, or “H”.

Note.— If not specified by the CM-ground-user, this indicates that there is no routing preference.

2.1.3.7.8 Result

Note.— This parameter indicates whether or not the information was forwarded as requested.

2.1.3.7.8.1 The *Result* parameter shall conform to the ASN.1 abstract syntax CMForwardResponse.

Note.— When the sending CM-ground-ASE version number is less than or equal to the receiving CM-ground-ASE version number the Result *parameter takes the abstract value “success”. When the sending CM-ground-ASE version number is greater than the receiving CM-ground-ASE version number the* Result *parameter takes the abstract value “incompatible version”. When the receiving CM-ground-ASE does not support ground-ground forwarding the* Result *parameter takes the abstract value “service not supported”.*

2.1.3.8 CM-user-abort Service

Note 1.— This service provides the capability for either the CM-air-user or a CM-ground-user to abort communication with its peer. This can be used for operational or technical reasons. It can be invoked at any time by an active user. Messages in transit may be lost during this operation. It is an unconfirmed service.

Note 2.— If the service is invoked prior to complete establishment of the dialogue, the CM-user-abort indication may not be provided. A CM-provider-abort indication may result instead.

2.1.3.8.1 The CM-user-abort service shall contain the primitives as presented Table 2.1.3-8.

Table 2.1.3-8. CM-user-abort Service Parameters

Parameter Name	Req	Ind
none		

2.1.3.9 CM-provider-abort Service

Note.— This service provides the capability for the CM-service provider to inform its users that it can no longer provide the CM service. Messages in transit may be lost during this operation.

2.1.3.9.1 The CM-provider-abort service shall contain the primitives and parameters as presented Table 2.1.3-9.

Table 2.1.3-9. CM-provider-abort Service Parameters

Parameter Name	Ind
Reason	M

2.1.3.9.2 Reason

Note.— This parameter identifies the reason for the abort.

2.1.3.9.2.1 The *Reason* parameter value shall conform to the ASN.1 abstract syntax CMAbortReason.

2.1.4 FORMAL DEFINITIONS OF MESSAGES

2.1.4.1 Encoding/decoding Rules

2.1.4.1.1 A CM-air-ASE shall be capable of encoding CMAircraftMessage APDUs and decoding CMGroundMessage APDUs.

2.1.4.1.2 A CM-ground-ASE shall be capable of encoding and decoding CMGroundMessage APDUs and decoding CMAircraftMessage APDUs.

2.1.4.2 CM ASN.1 Abstract Syntax

2.1.4.2.1 The abstract syntax of the CM protocol data units shall comply with the description contained in the ASN.1 module CMMessageSetVersion1 (conforming to ISO/IEC 8824-1), as defined in 2.1.4.

```
CMMessageSetVersion1 DEFINITIONS ::=

BEGIN

-- ------------------------------------------------------------------------------
-- CM Message Structure
-- ------------------------------------------------------------------------------
--  Aircraft-generated messages

CMAircraftMessage ::=CHOICE
        {
        cmLogonRequest          [0]     CMLogonRequest,
        cmContactResponse       [1]     CMContactResponse,
        cmAbortReason           [2]     CMAbortReason,
        ...
        }

--  Ground-generated messages

CMGroundMessage ::=CHOICE
        {
        cmLogonResponse         [0]     CMLogonResponse,
        cmUpdate                [1]     CMUpdate,
        cmContactRequest        [2]     CMContactRequest,
        cmForwardRequest        [3]     CMForwardRequest,
        cmAbortReason           [4]     CMAbortReason,
        cmForwardResponse       [5]     CMForwardResponse,
        ...
        }
```

```
-- ------------------------------------------------------------------------------
-- CM Message Components
-- ------------------------------------------------------------------------------

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

Airport ::=IA5String (SIZE(4))

APAddress ::= CHOICE
        {
        longTsap        [0]     LongTsap,
        shortTsap       [1]     ShortTsap
        }

AEQualifier ::= INTEGER (0..255)
-- ATN AE-Qualifier Numeric Values are described in 4

AEQualifierVersion ::= SEQUENCE
        {
        aeQualifier             AEQualifier,
        apVersion               VersionNumber
        }

AEQualifierVersionAddress ::= SEQUENCE
        {
        aeQualifier             AEQualifier,
        apVersion               VersionNumber,
        apAddress               APAddress
        }

CMAbortReason ::= ENUMERATED
        {
        timer-expired                           (0),
        undefined-error                 (1),
        invalid-PDU                             (2),
        not-permitted-PDU                       (3),
        dialogue-acceptance-not-permitted       (4),
        dialogue-end-not-accepted               (5),
        communication-service-error             (6),
        communication-service-failure           (7),
        invalid-QOS-parameter                   (8),
        expected-PDU-missing                    (9),
        ...
        }
```

```
CMContactRequest ::= SEQUENCE
        {
        facilityDesignation             FacilityDesignation,
        address                         LongTsap
        }

CMContactResponse ::= Response

CMForwardRequest ::= CMLogonRequest

CMForwardResponse ::= ENUMERATED
        {
        success                         (0),
        incompatible-version            (1),
        service-not-supported           (2)
        }

CMLogonRequest ::= SEQUENCE
        {
        aircraftFlightIdentification    [0] AircraftFlightIdentification,
        cMLongTSAP                      [1] LongTsap,
        groundInitiatedApplications     [2] SEQUENCE SIZE (1..256) OF  AEQualifierVersionAddress
                                                                       OPTIONAL,
        airOnlyInitiatedApplications    [3] SEQUENCE SIZE (1..256) OF  AEQualifierVersion
                                                                       OPTIONAL,
        facilityDesignation             [4] FacilityDesignation        OPTIONAL,
        airportDeparture                [5] Airport                    OPTIONAL,
        airportDestination              [6] Airport                    OPTIONAL,
        dateTimeDepartureETD            [7] DateTime                   OPTIONAL
        }

CMLogonResponse ::= SEQUENCE
        {
        airInitiatedApplications        [0] SEQUENCE SIZE (1..256) OF  AEQualifierVersionAddress
                                                                       OPTIONAL,
        groundOnlyInitiatedApplications [1] SEQUENCE SIZE (1..256) OF  AEQualifierVersion
                                                                       OPTIONAL
        }

CMUpdate ::= CMLogonResponse

Date ::= SEQUENCE
        {
        year            Year,
        month           Month,
        day             Day
        }
```

```
        -- The Date field does not have to correspond to the flight if the field is not to be used;
        -- the field's value can be assigned a meaningless, but compliant, value locally.  If operational
        -- use of the Date field is intended, there must be bilateral agreements in place to ensure its proper
        -- use.  This is a local implementation issue.

DateTime ::= SEQUENCE
        {
        date            Date,
        time            Time
        }

Day ::= INTEGER (1..31)
        --unit = Day, Range (1..31), resolution = 1

FacilityDesignation ::= IA5String (SIZE(4..8))

LongTsap ::= SEQUENCE
        {
        rDP                     OCTET STRING (SIZE(5)),
        shortTsap               ShortTsap
        }

Month ::= INTEGER (1..12)
        --unit = Month, Range (1..12), resolution = 1

Response ::= ENUMERATED
        {
        contactSuccess                  (0),
        contactNotSuccessful            (1)
        }

ShortTsap ::= SEQUENCE
        {
        aRS             [0]     OCTET STRING (SIZE(3))          OPTIONAL,
        -- the aRS contains the ICAO 24 bit aircraft address when the ShortTsap belongs to an aircraft;
        -- or a ground address when the Short Tsap belongs to a ground system
        locSysNselTsel  [1]     OCTET STRING  (SIZE(10..11))
        }

Time ::= SEQUENCE
        {
        hours                   Timehours,
        minutes                 Timeminutes
        }

Timehours ::= INTEGER (0..23)
        -- units = hour, range (0..23), resolution = 1 hour
```

Timeminutes ::= INTEGER (0..59)
-- units = minute, range (0..59), resolution = 1 minute

VersionNumber ::= INTEGER (1..255)
-- VersionNumber 0 is reserved for the Dialogue Service

Year ::= INTEGER (1996..2095)
--unit = Year, Range (1996..2095), resolution = 1

END

2.1.5 PROTOCOL DEFINITION

2.1.5.1 Sequence Rules

2.1.5.1.1 With the exception of abort primitives, only the sequence of primitives described in figures 2.1.5-1 through 2.1.5-22 shall be permitted.

Note 1.— The following figures define the valid sequences of primitives that are possible to be invoked during the operation of the CM application. It shows the relationship in time between the service request and the resulting indication, and if applicable, the subsequent response and resulting confirmation.

Note 2.— Abort primitives may interrupt and terminate any of the normal message sequences outlined below.

Note 3.— More than one CM-logon attempt may be made for a given CM-contact request. The number of attempts may be determined by local procedures.

Note 4.— Primitives are processed in the order in which they are received. See 4.3.3.

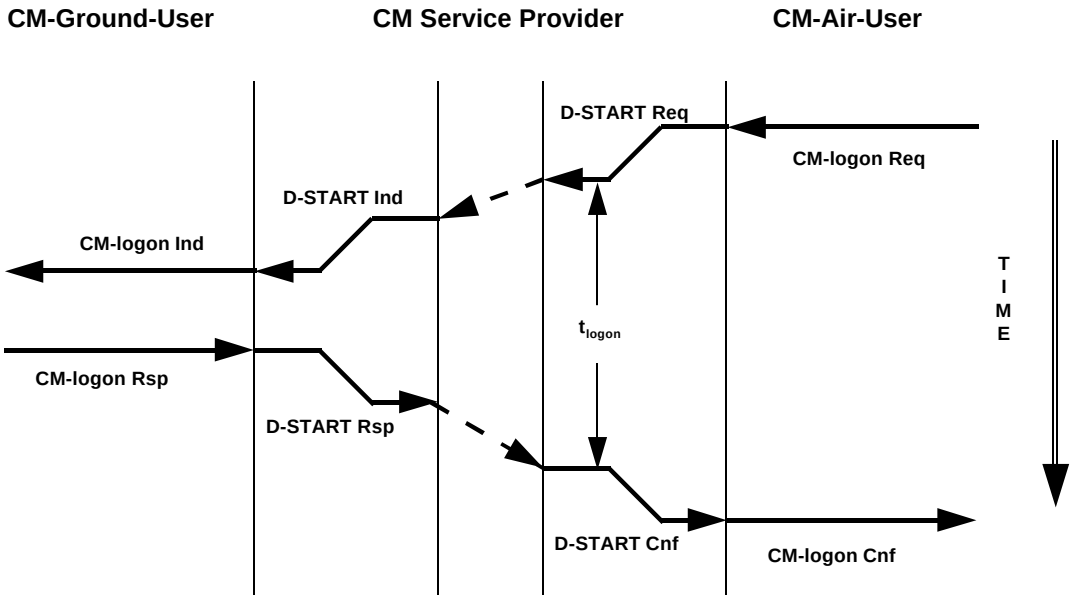

Figure 2.1.5-1. Sequence Diagram for CM-logon Service
CM-Air-ASE Version ≤ CM-Ground-ASE Version

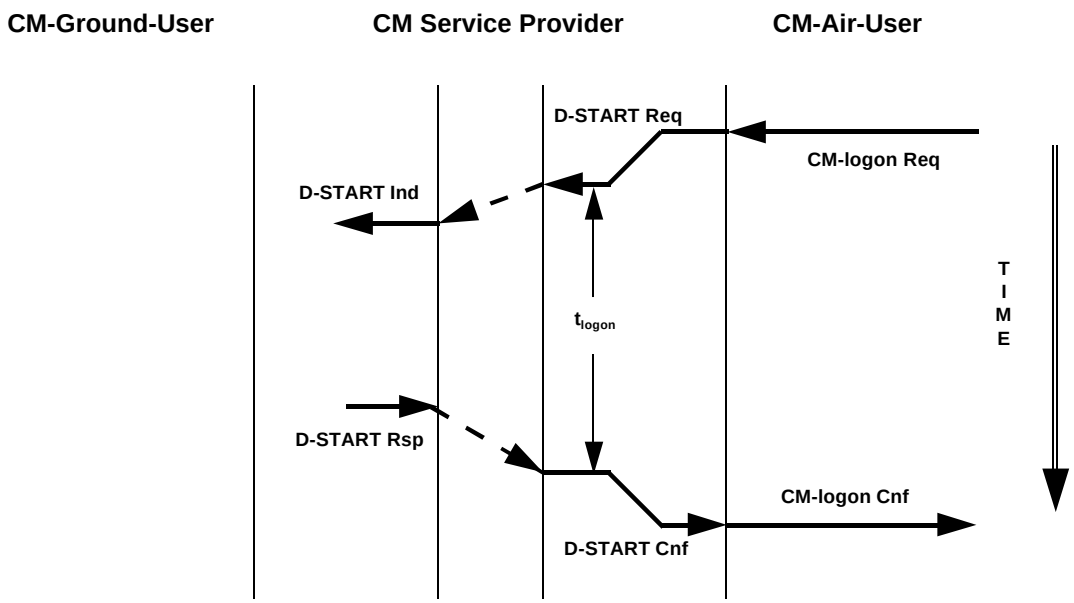

Figure 2.1.5-2. Sequence Diagram for CM-logon Service
CM-Air-ASE Version >CM-Ground-ASE Version

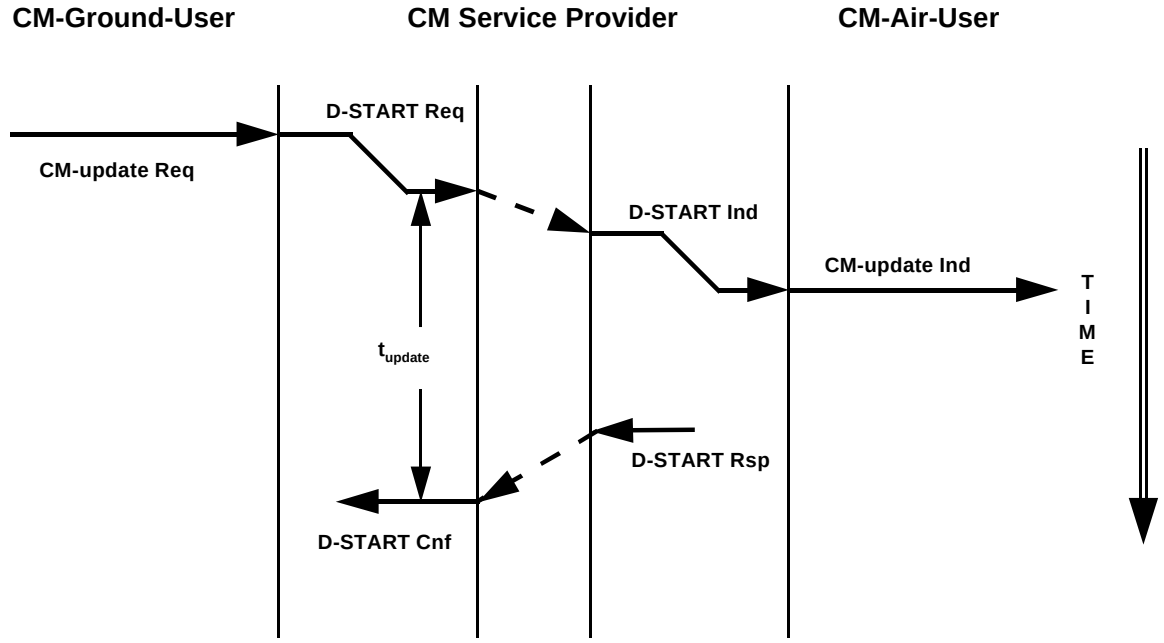

Figure 2.1.5-3. Sequence Diagram for CM-update Service
No Existing CM Dialogue

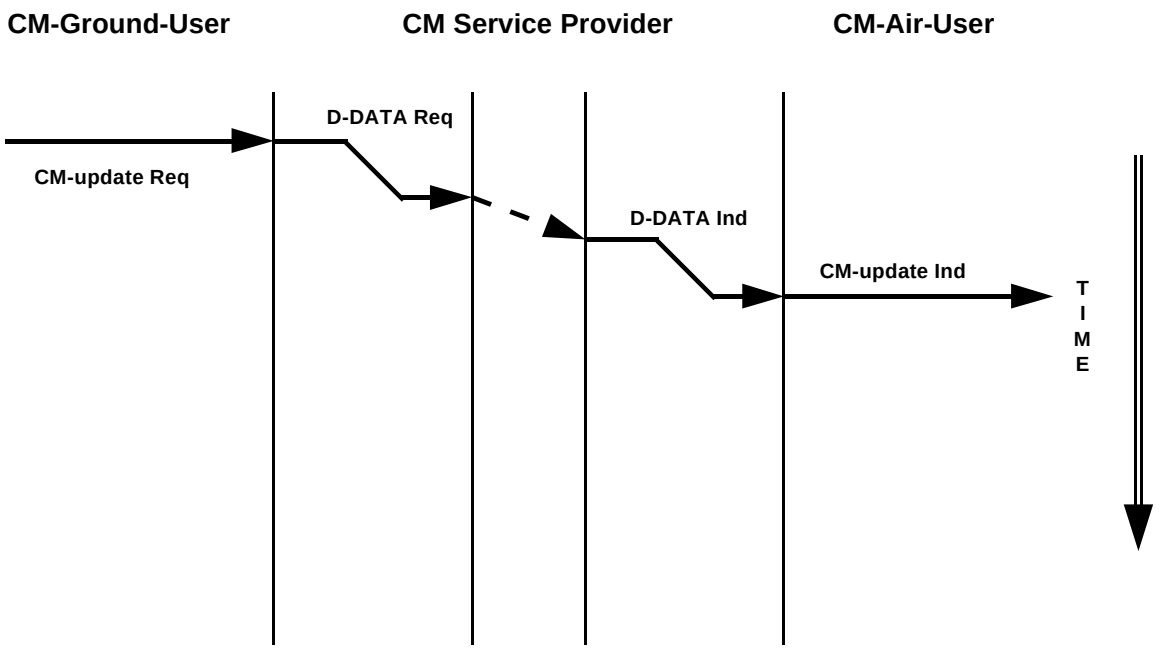

Figure 2.1.5-4. Sequence Diagram for CM-update Service Existing CM Dialogue

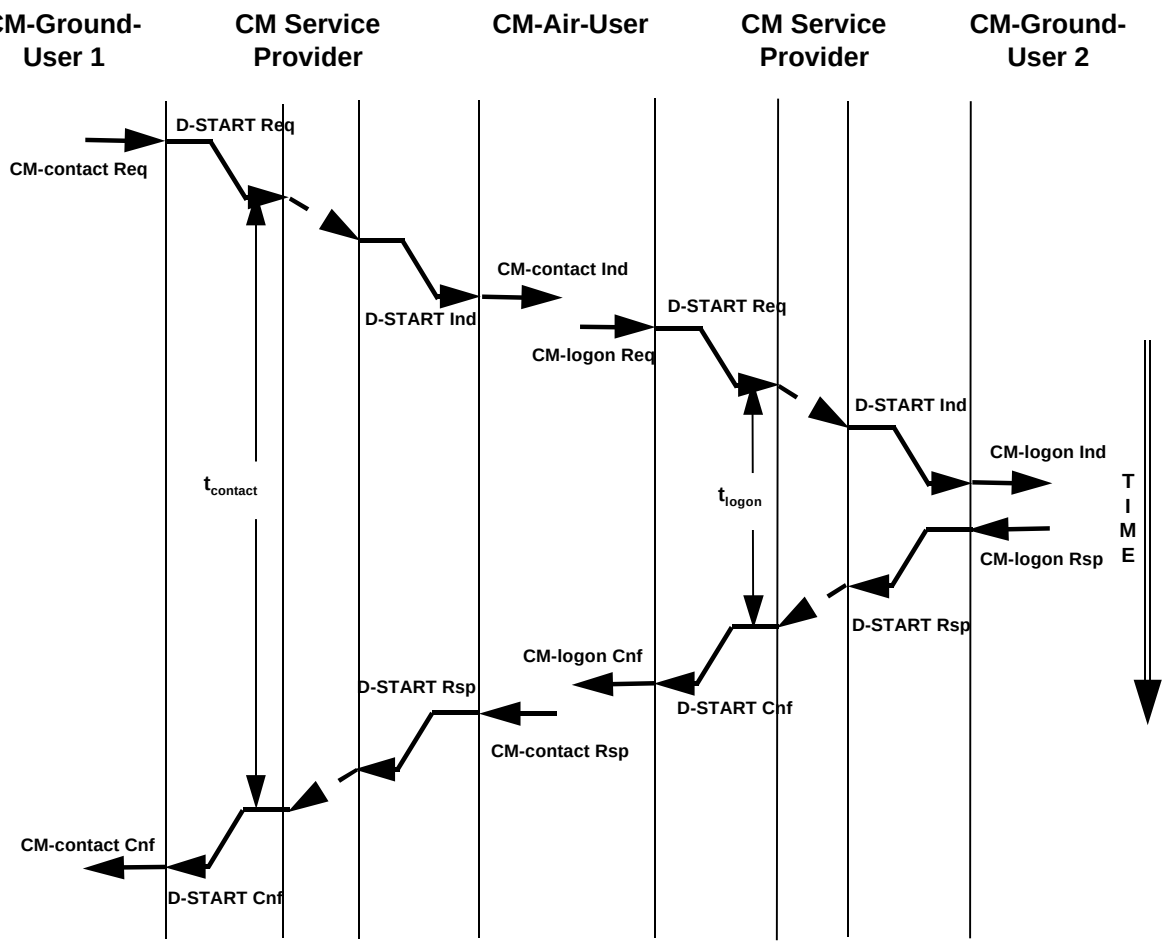

Figure 2.1.5-5. Sequence Diagram for CM-contact Service
No Existing CM Dialogue
With CM-logon Service as in Figure 2.1.5-1

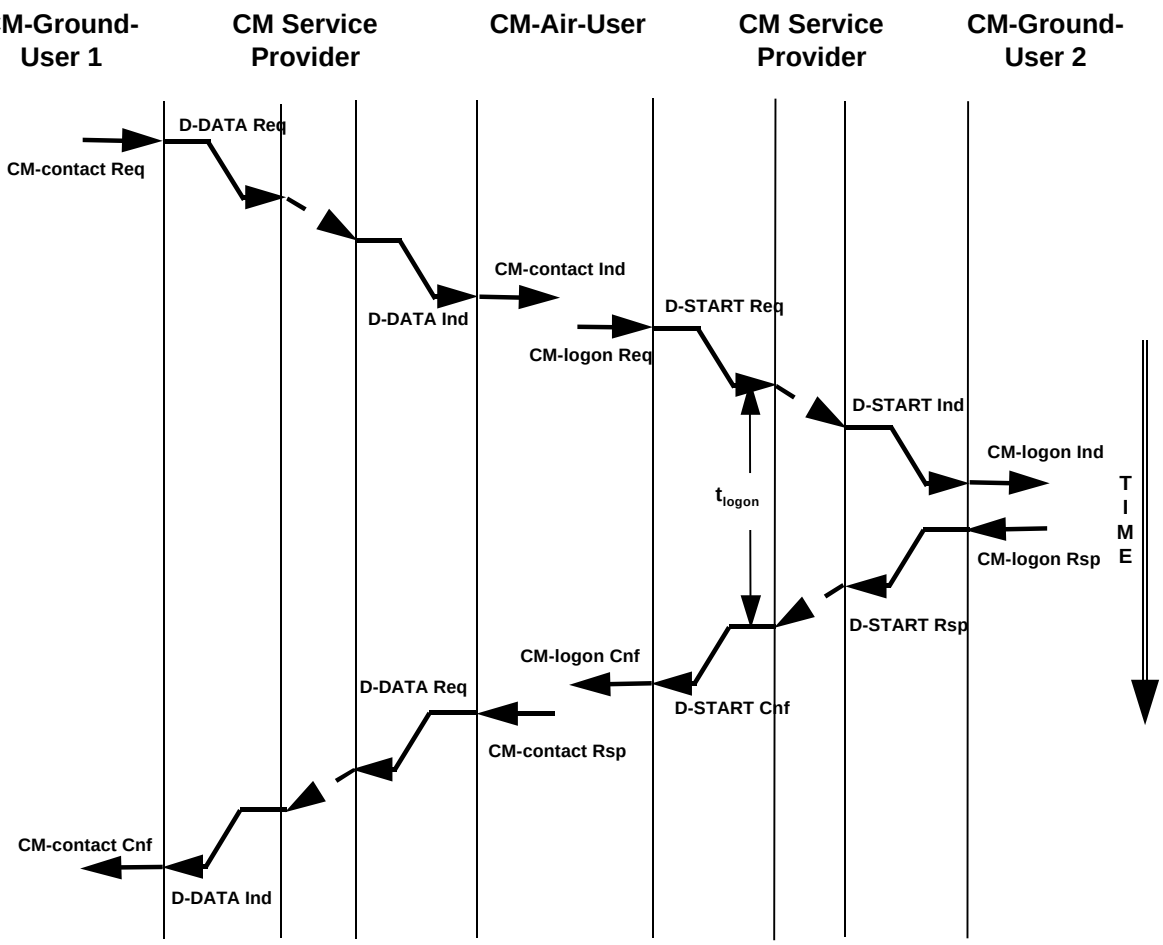

Figure 2.1.5-6. Sequence Diagram for CM-contact Service
With Existing CM Dialogue
With CM-logon Service as in Figure 2.1.5-1

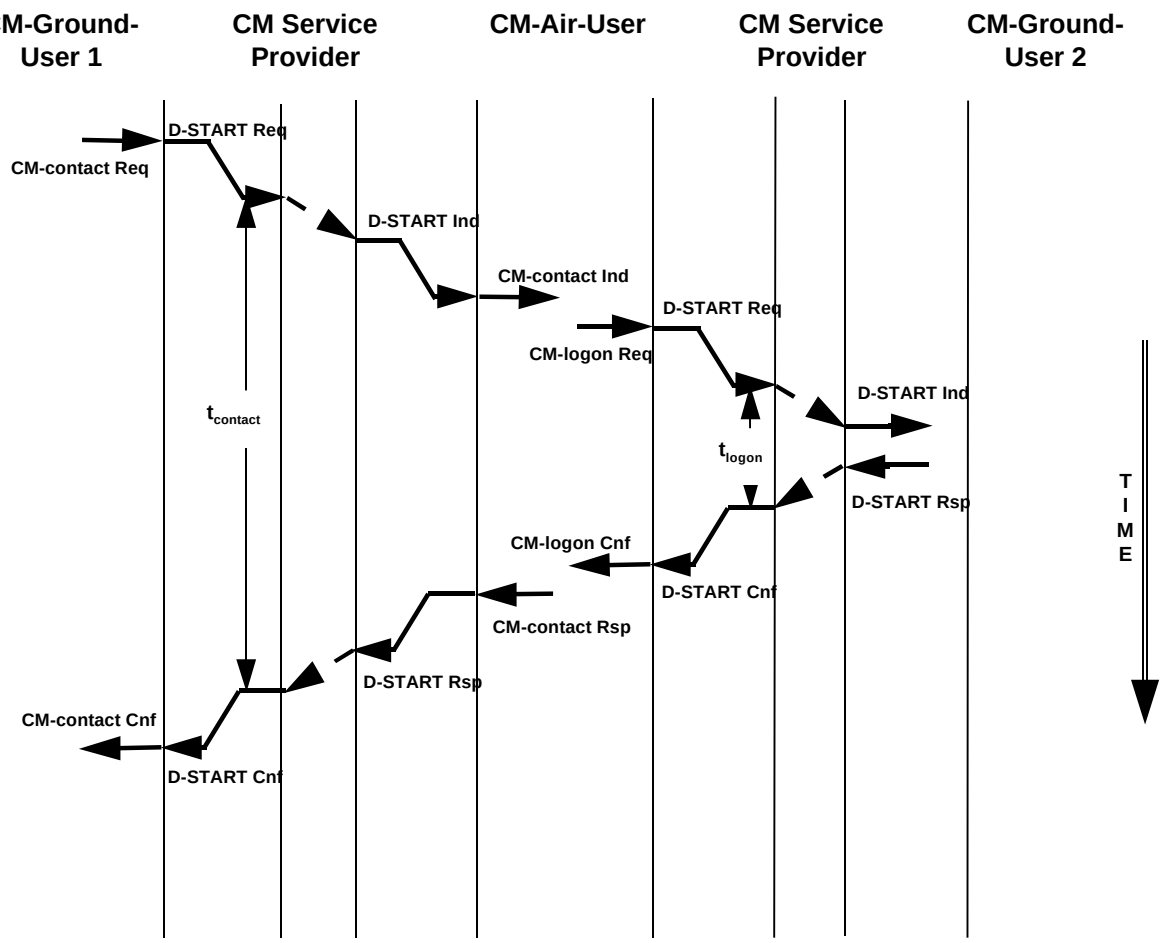

Figure 2.1.5-7. Sequence Diagram for CM-contact Service
No Existing CM Dialogue
With CM-logon Service as in Figure 2.1.5-2

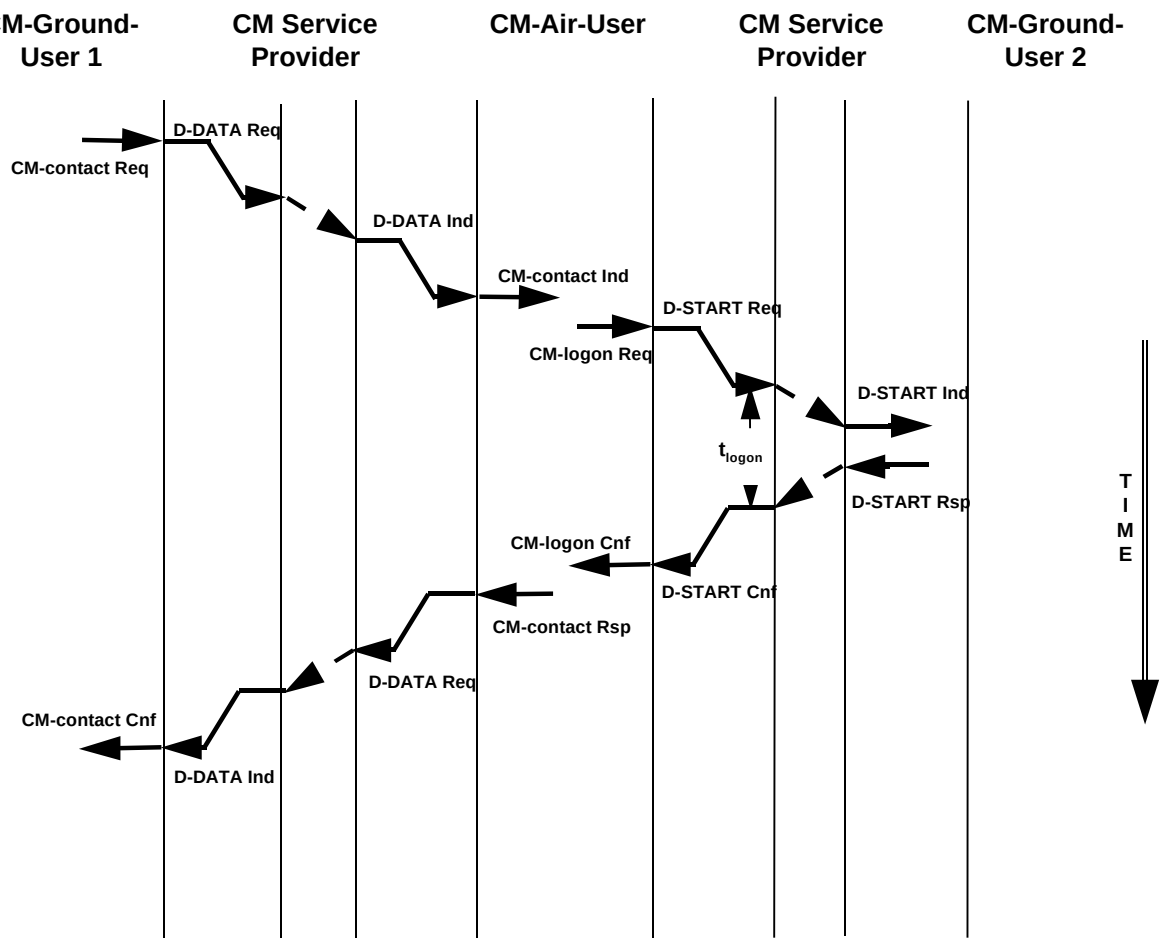

Figure 2.1.5-8. Sequence Diagram for CM-contact Service
Existing CM Dialogue
With CM-logon Service as in Figure 2.1.5-2

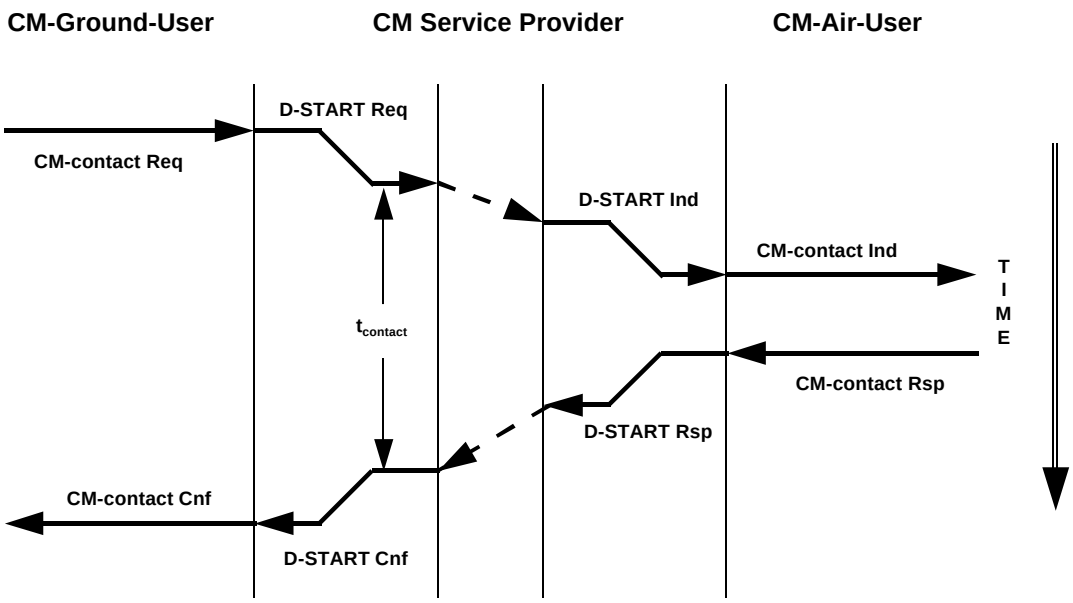

Figure 2.1.5-9. Sequence Diagram for CM-contact Service
No Existing CM Dialogue
Without CM-logon Service as Requested

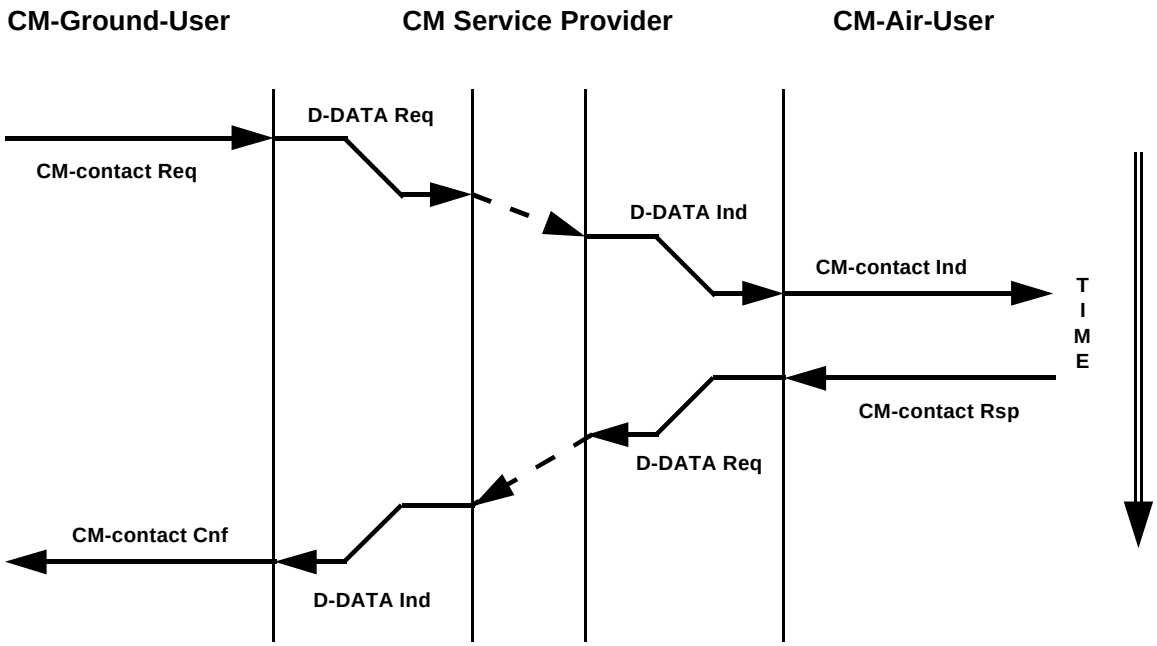

Figure 2.1.5-10. Sequence Diagram for CM-contact Service With Existing CM Dialogue Without CM-logon Service as Requested

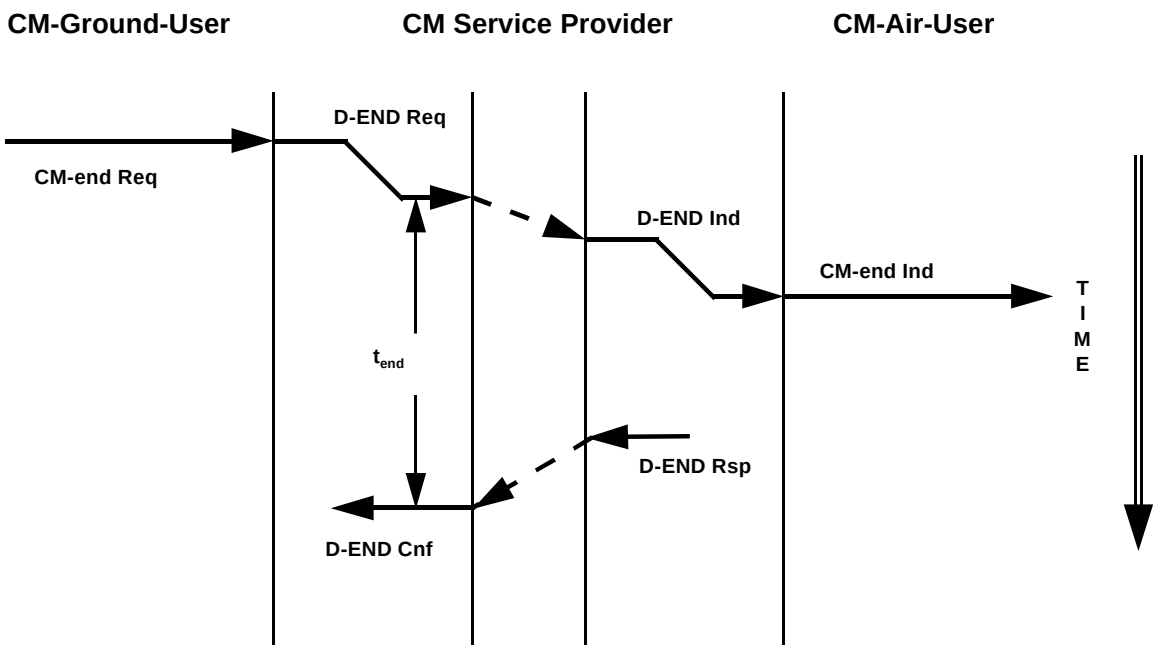

Figure 2.1.5-11. Sequence Diagram for CM-end Service

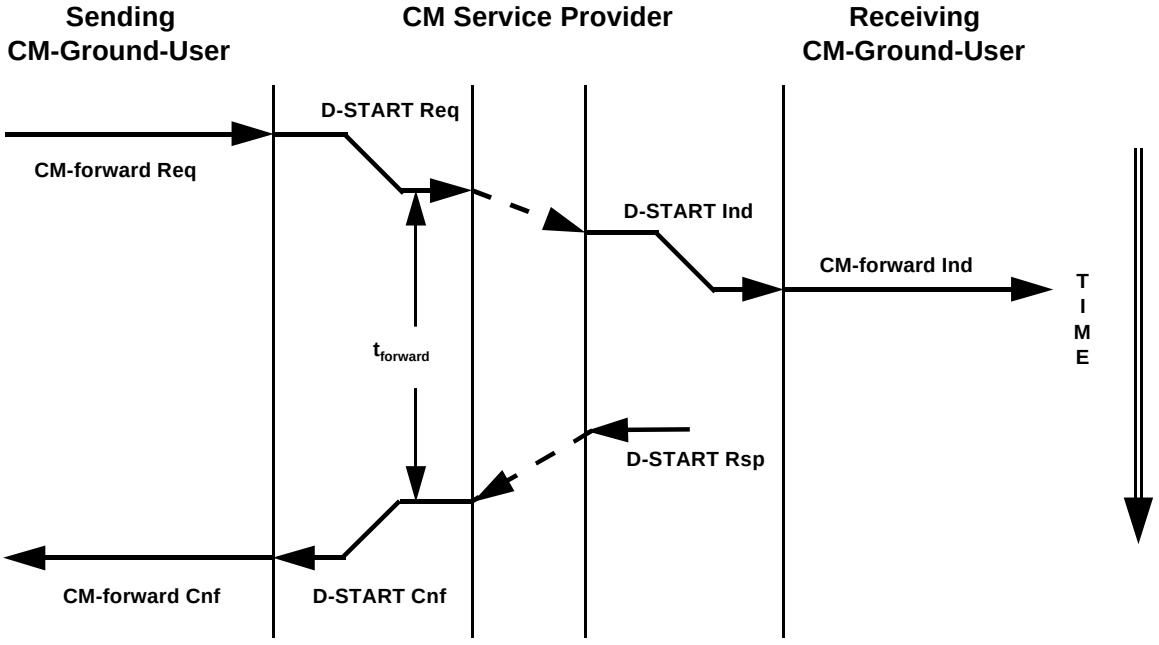

Figure 2.1.5-12. Sequence Diagram for CM-forward Service
Sending CM-Ground-ASE Version ≤ Receiving CM-Ground-ASE Version,
Ground-ground Forwarding Supported

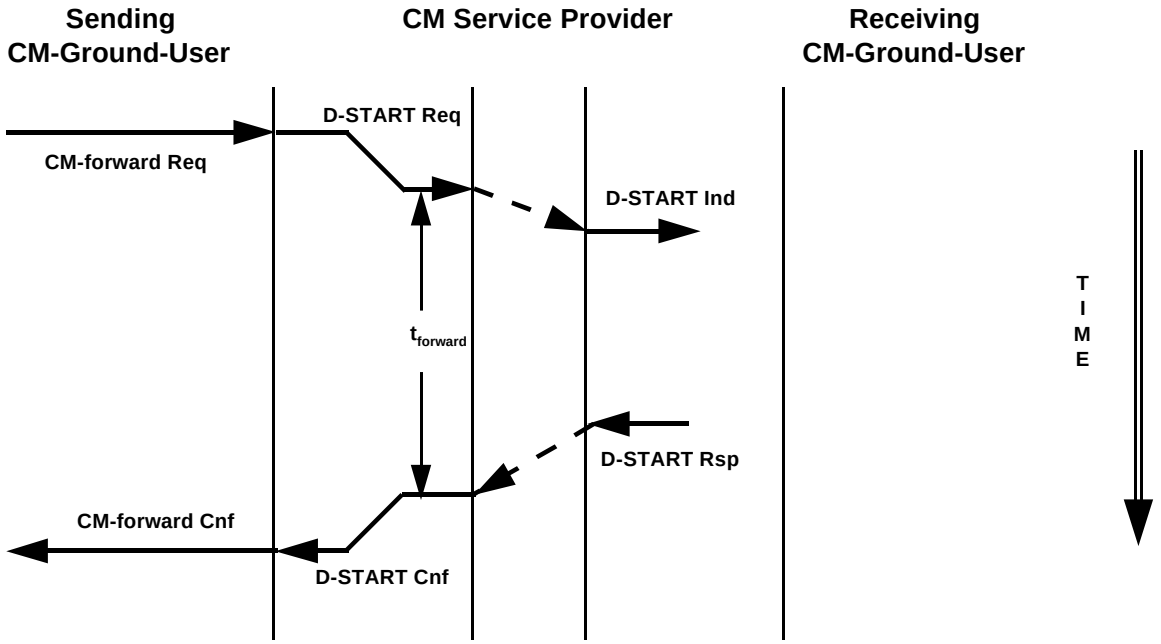

Figure 2.1.5-13. Sequence Diagram for CM-forward Service
Sending CM-Ground-ASE Version >Receiving CM-Ground-ASE Version
or Receiving CM-Ground-ASE does not Support Ground-Ground Forwarding

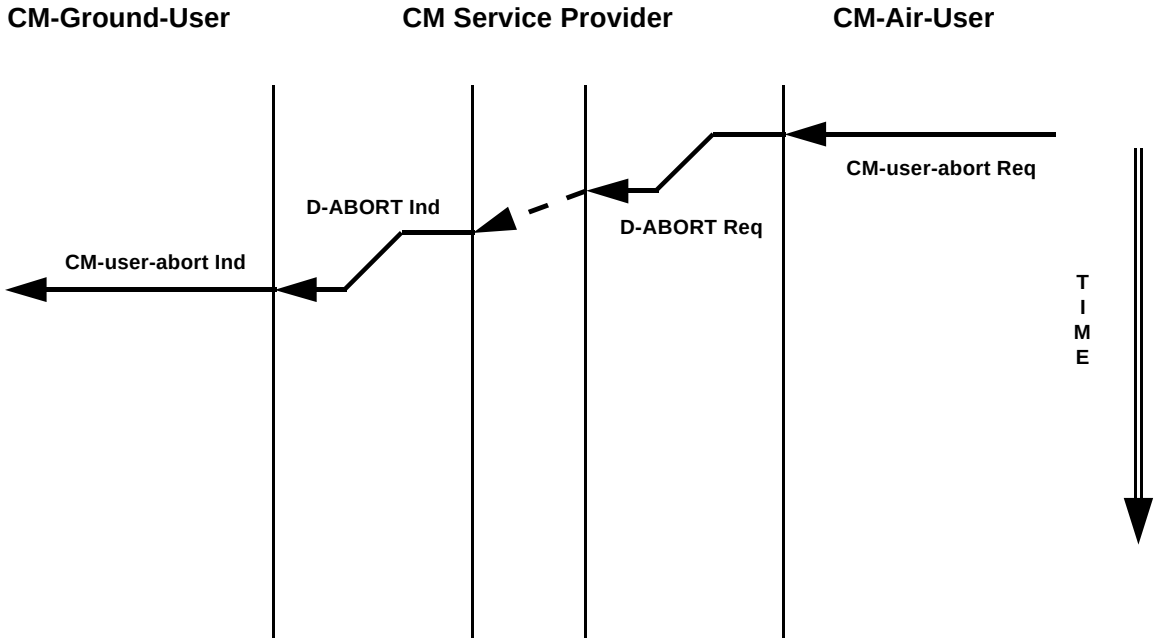

Figure 2.1.5-14. Sequence Diagram for CM-user-abort Service
CM-Air-User Initiated

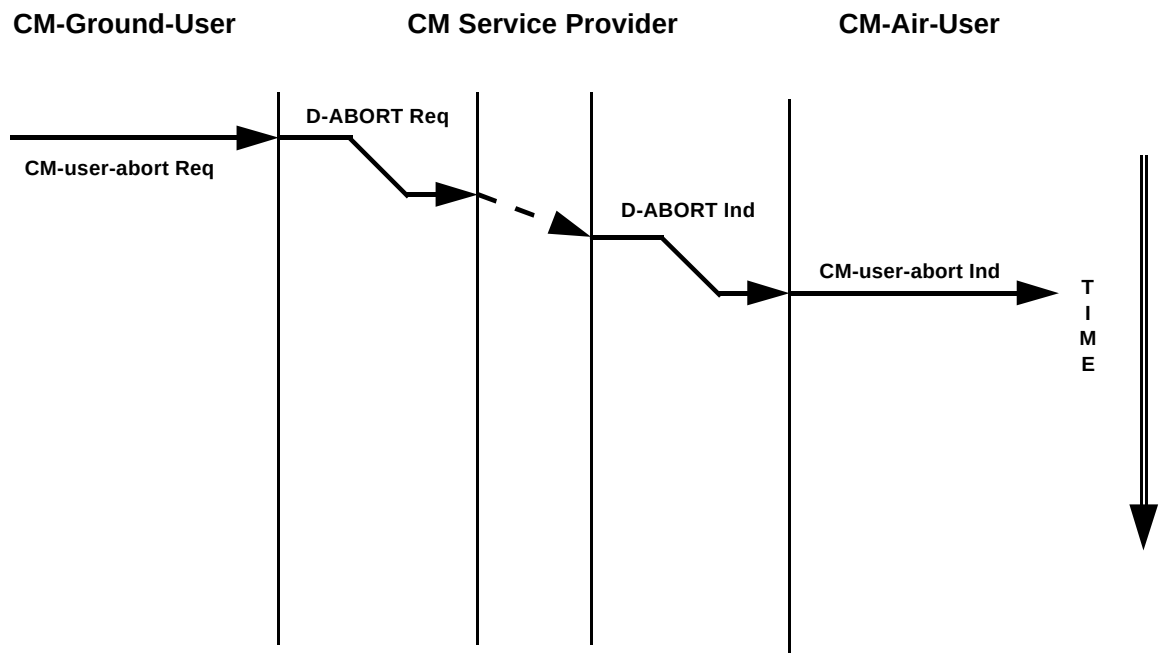

Figure 2.1.5-15. Sequence Diagram for CM-user-abort Service CM-Ground-User Initiated

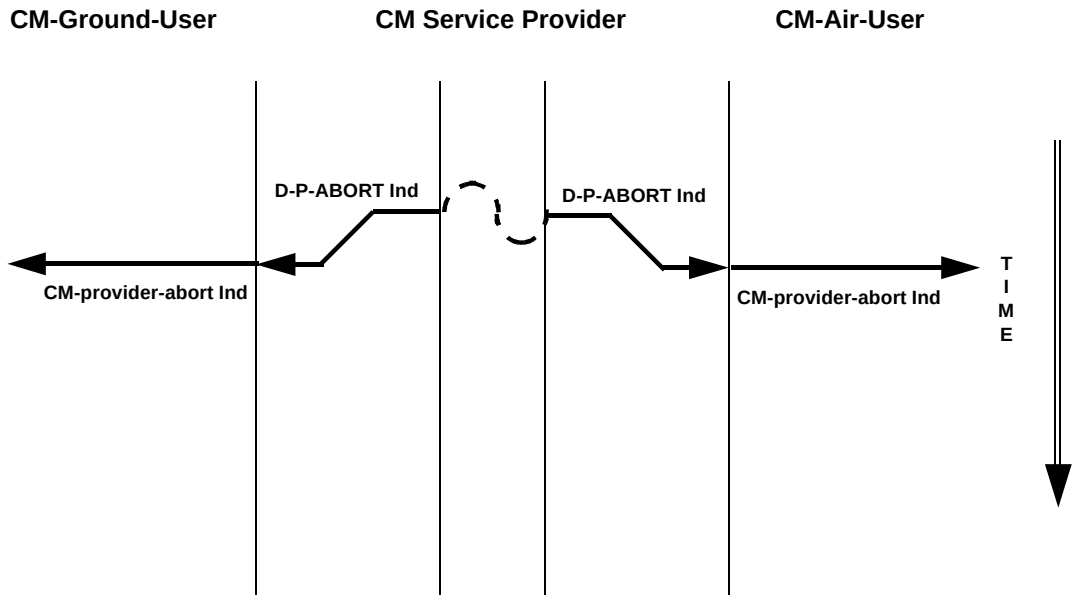

Figure 2.1.5-16. Sequence Diagram for CM-provider-abort Service: Dialogue Service Abort

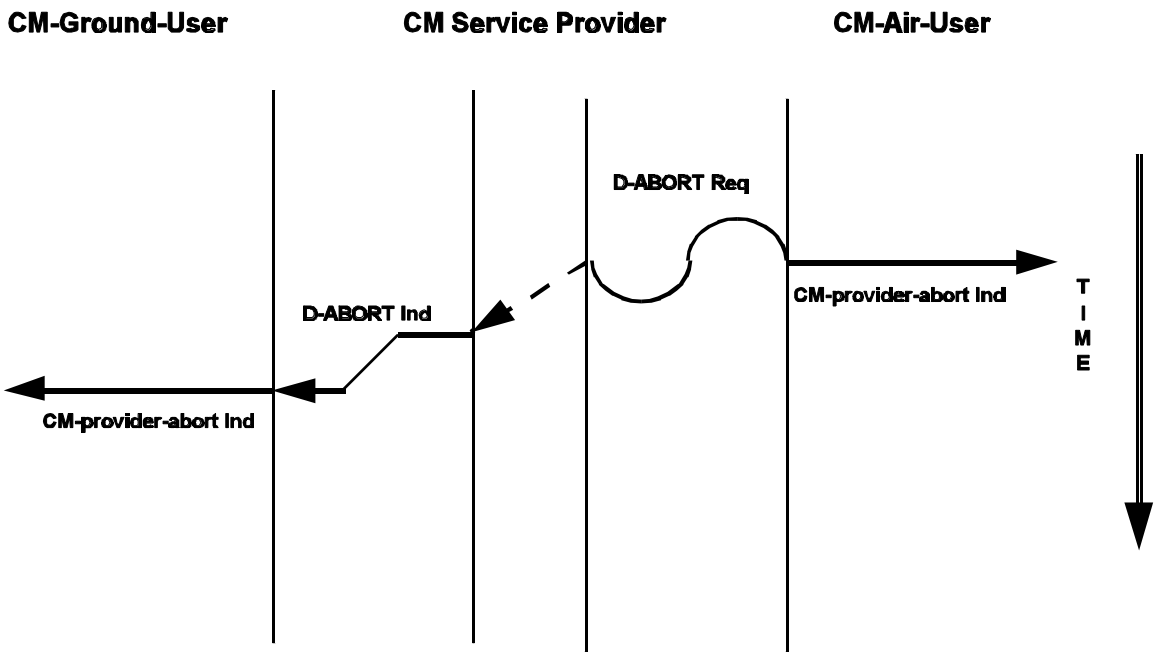

Figure 2.1.5-17. Sequence Diagram for CM-provider-abort Service: CM-Air-ASE Abort

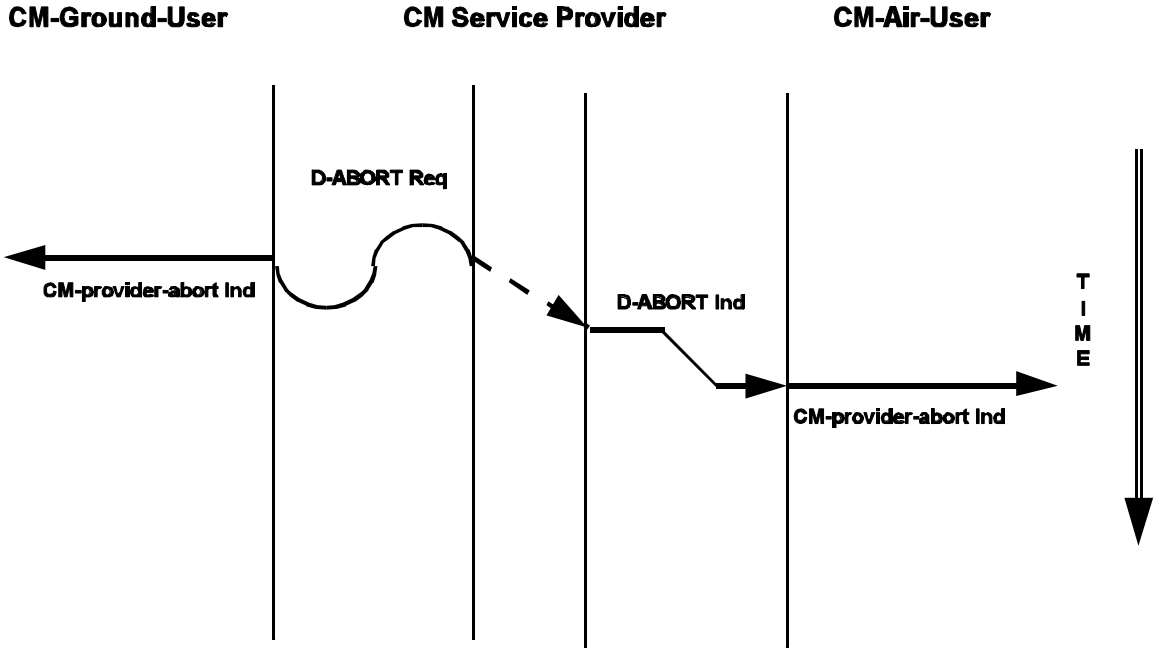

Figure 2.1.5-18. Sequence Diagram for CM-provider-abort Service: CM-Ground-ASE Abort

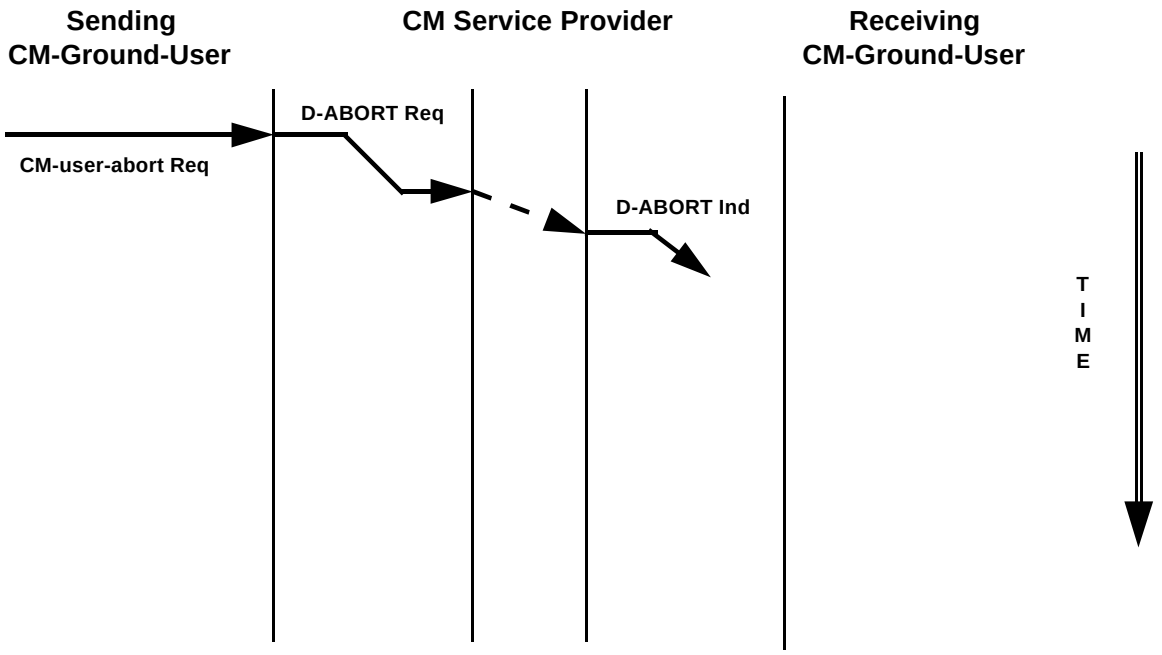

Figure 2.1.5-19. Sequence Diagram for CM-user-abort Service Sending CM-Ground-User Initiated

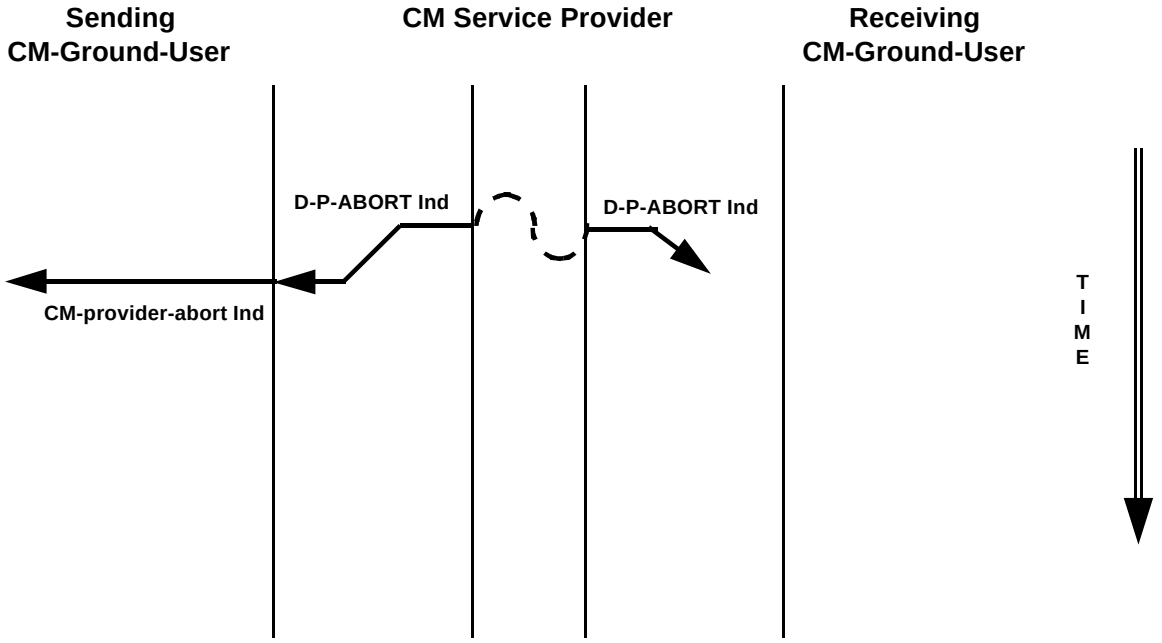

Figure 2.1.5-20. Sequence Diagram for CM-provider-abort Service: Dialogue Service Abort

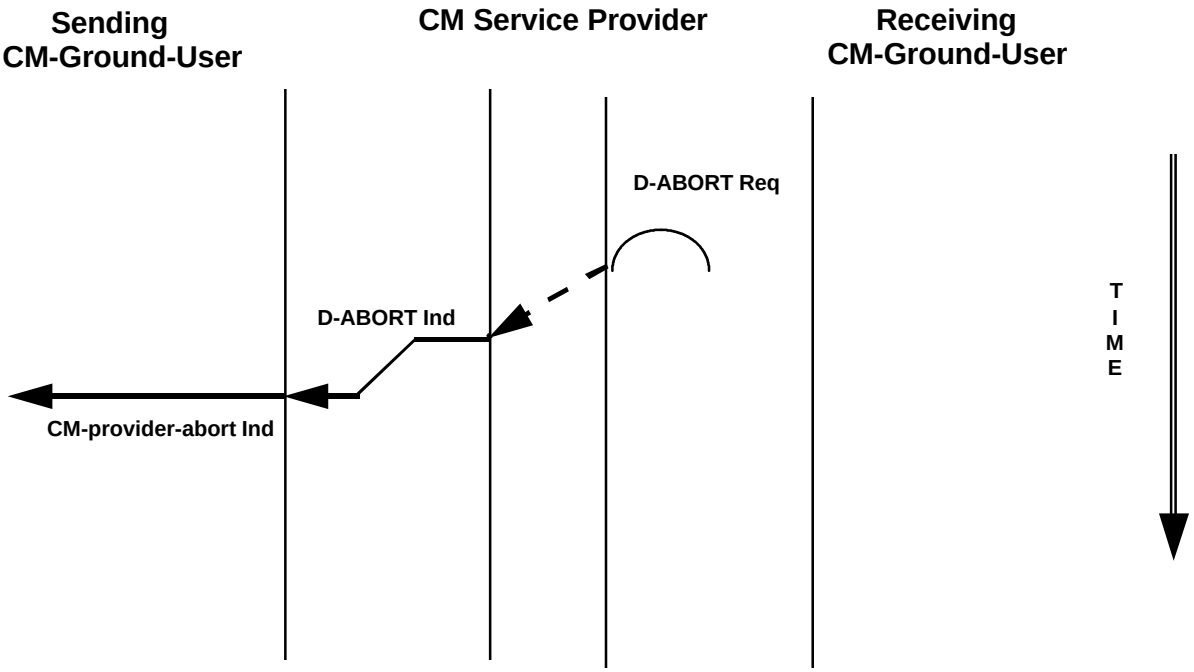

Figure 2.1.5-21. Sequence Diagram for CM-provider-abort Service: Receiving CM-Ground-ASE Abort

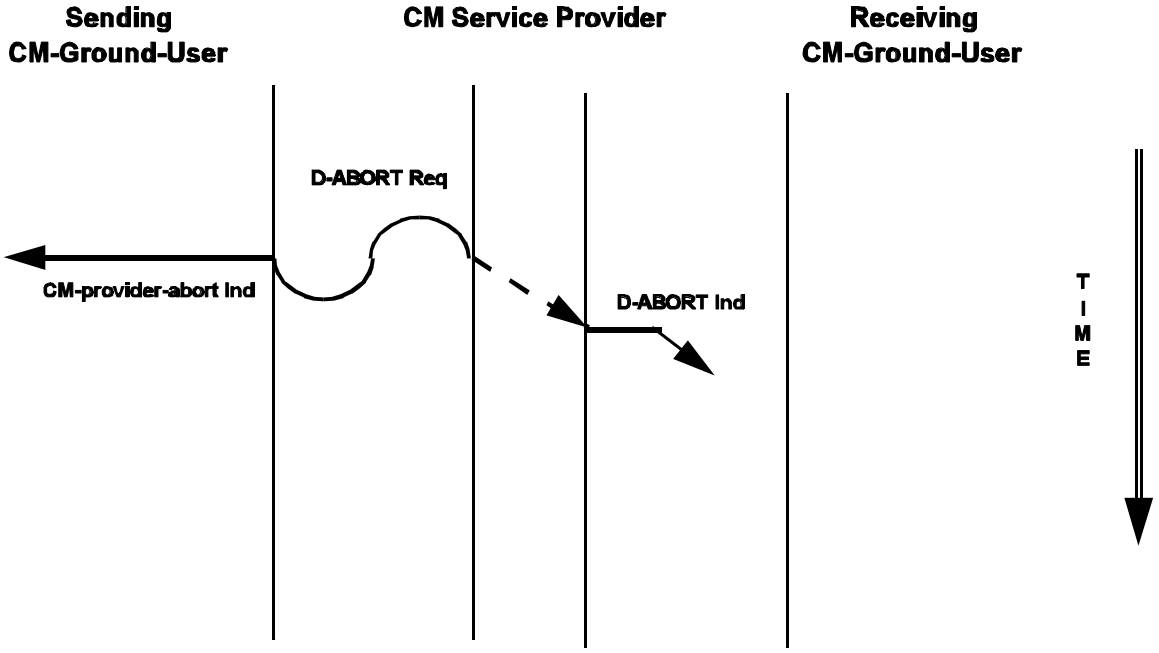

Figure 2.1.5-22. Sequence Diagram for CM-provider-abort Service: Sending CM-Ground-ASE Abort

2.1.5.2 CM Service Provider Timers

2.1.5.2.1 A CM-ASE shall be capable of detecting when a timer expires.

Note 1.— Table 2.1.5-1 lists the time constraints related to the CM application. Each time constraint requires a timer to be set in the CM protocol machine.

Note 2.— If the timer expires before the final event has occurred, a CM-ASE takes the appropriate action as defined in 2.1.5.4.1.

2.1.5.2.2 **Recommendation.** — *The timer values should be as indicated in Table 2.1.5-1.*

Table 2.1.5-1. CM Service Provider Timers

CM Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
CM-logon	t_{logon}	4 min	D-START request	D-START confirmation
CM-update	t_{update}	4 min	D-START request	D-START confirmation
CM-contact	$t_{contact}$	8 min	D-START request	D-START confirmation
CM-forward	$t_{forward}$	4 min	D-START request	D-START confirmation
CM-end	t_{end}	4 min	D-END request	D-END confirmation

Note.— The receipt of a CM-user-abort request, D-ABORT indication, or D-P-ABORT indication are also timer stop events.

2.1.5.3 CM-ASE Protocol Description

2.1.5.3.1 Introduction

Note.— 2.1.5.3 presents requirements for CM-ASEs in specific states. 2.1.5.5 contains state tables for the CM-ASEs.

2.1.5.3.1.1 If no actions are described for a CM service primitive when a CM-ASE is in a specific state, then the invocation of that service primitive shall be prohibited while the CM-ASE is in that state.

2.1.5.3.1.2 Upon receipt of a PDU, if no actions are described for the arrival of that PDU when a CM-ASE is in a specific state, then that PDU is considered not permitted and exception handling procedures as described in 2.1.5.4.4 shall apply.

2.1.5.3.1.3 If a PDU is received that cannot be decoded, then that PDU is considered an invalid PDU and exception handling procedures as described in 2.1.5.4.3 shall apply.

2.1.5.3.1.4 If a PDU is not received where one is expected, then that PDU is considered an invalid PDU and exception handling procedures as described in 2.1.5.4.3 shall apply.

2.1.5.3.2 CM-Air-ASE Protocol Description

Note 1.— The states defined for the CM-air-ASE are the following:

a) IDLE,

b) LOGON,

c) CONTACT,

d) DIALOGUE, and

e) CONTACT DIALOGUE.

Note 2.— The CM-air-user is considered an active user from the time:

a) the CM-air-user invokes a CM-logon Req until it:

1) receives a CM-logon Cnf, if a dialogue is not maintained,

2) receives a CM-end Req, if a dialogue is maintained,

3) receives a CM-user abort,

4) receives a CM-provider abort, or

5) invokes a CM-user abort,

b) the CM-air-user receives a CM-contact Ind until it:

1) invokes a CM-contact Rsp, if a dialogue is not maintained,

2) receives a CM-user abort,

3) receives a CM-provider abort, or

4) invokes a CM-user abort.

2.1.5.3.2.1 On initiation, the CM-air-ASE shall be in the *IDLE* state.

2.1.5.3.2.2 D-START Indication

2.1.5.3.2.2.1 Upon receipt of a D-START indication, if the CM-air-ASE is in the *IDLE* state and the D-START indication *Priority* Quality-of-Service parameter has the abstract value "flight regularity communications" and the *RER* Quality-of-Service parameter has the abstract value of "low" then:

2.1.5.3.2.2.1.1 If the APDU contained in the D-START *User Data* parameter is a [CMUpdate] APDU the CM-air-ASE shall:

a) invoke CM-update service indication with the following:

1) the D-START Calling Peer ID parameter value as the CM-update *Facility Designation* parameter value, and

2) the APDU contained in the D-START *User Data* parameter as the CM-update *Update Information* parameter value,

b) invoke D-START response with the abstract value "rejected (permanent)" provided as D-START *Result* parameter value, and

c) enter the *IDLE* state.

2.1.5.3.2.2.1.2 If the APDU contained in the D-START *User Data* parameter is a [CMContactRequest] APDU the CM-air-ASE shall:

a) invoke CM-contact service indication with the following:

1) The D-START *Calling Peer ID* parameter value as the CM-contact *Facility Designation* parameter value, and

2) The APDU contained in the D-START *User Data* parameter as the CM-contact *Contact Request* parameter value, and

b) enter the *CONTACT* state.

2.1.5.3.2.3 D-START Confirmation

2.1.5.3.2.3.1 Upon receipt of a D-START confirmation, if the CM-air-ASE is in the *LOGON* state and if the APDU contained in the D-START *User Data* parameter is a [CMLogonResponse] APDU or if the D-START *User Data* parameter is not present but the D-START *DS User Version Number* parameter is present, the CM-air-ASE shall:

a) stop timer t_{logon},

b) if the abstract value of the D-START *Result* parameter is "rejected (permanent)" and the abstract value of the D-START *Reject Source* parameter is "DS user" then:

1) if the version number of the CM-air-ASE is greater than the D-START *DS User Version Number* parameter value, invoke CM-logon service confirmation with the D-START *DS User Version Number* parameter value as the CM-logon *CM-ASE Version Number* parameter value, or

2) else invoke CM-logon service confirmation with the APDU contained in the D-START *User Data* parameter as the CM-logon *Logon Response* parameter, and

3) enter the *IDLE* state.

c) if the abstract value of the D-START *Result* parameter is "accepted" then:

1) invoke CM-logon service confirmation with the following:

i) the APDU contained in the D-START *User Data* parameter as the CM-logon *Logon Response* parameter value,

ii) the D-START *Result* parameter as the CM-Logon *Maintain Dialogue* parameter value, and

2) enter the *DIALOGUE* state.

2.1.5.3.2.4 D-DATA Indication

2.1.5.3.2.4.1 Upon receipt of a D-DATA indication, if the CM-air-ASE is in the *DIALOGUE* state then:

2.1.5.3.2.4.1.1 If the APDU contained in the D-DATA *User Data* parameter is a [CMUpdate] APDU the CM-air-ASE shall:

a) invoke CM-update service indication with the APDU contained in the D-DATA *User Data* parameter as the CM-update service *Update Information* parameter value, and

b) remain in the *DIALOGUE* state.

2.1.5.3.2.4.1.2 If the APDU contained in the D-DATA *User Data* parameter is a [CMContactRequest] APDU the CM-air-ASE shall:

a) invoke CM-contact service indication with the APDU contained in the D-DATA *User Data* parameter as the CM-contact service *Contact Request* parameter value, and

b) enter the *CONTACT DIALOGUE* state.

2.1.5.3.2.5 D-END Indication

2.1.5.3.2.5.1 Upon receipt of a D-END indication, if the CM-air-ASE is in the *DIALOGUE* state then the CM-air-ASE shall:

a) invoke CM-end service indication,

b) invoke D-END response with the D-END *Result* parameter set to the abstract value "accepted", and

c) enter the *IDLE* state.

2.1.5.3.2.6 CM-logon Service Request

2.1.5.3.2.6.1 Upon receipt of a CM-logon service request:

2.1.5.3.2.6.1.1 If the CM-air-ASE is in the *IDLE* state the CM-air-ASE shall:

a) create a CMAircraftMessage APDU with a cmLogonRequest APDU-element based on the *Logon Request* parameter value,

b) invoke D-START request with the following:

1) the CM-logon *Facility Designation* parameter value as the D-START *Called Peer ID* parameter value,

2) the CM-logon *Aircraft Address* parameter value as the D-START *Calling Peer ID* parameter value,

3) the CM-air-ASE version number as the D-START *DS User Version Number* parameter value,

4) the D-START *Quality of Service* parameters set as follows:

i) if provided, the CM-logon service *Class of Communication Service* parameter value as the D-START *Routing Class* parameter value,

ii) the abstract value of "flight regularity communications", as the D-START *Priority* parameter value, and

iii) the abstract value of "low" as the D-START *RER* parameter value, and

5) the CMAircraftMessage APDU as the D-START *User Data* parameter value,

c) start timer t_{logon}, and

d) enter the *LOGON* state.

2.1.5.3.2.7 CM-contact Service Response

2.1.5.3.2.7.1 Upon receipt of a CM-contact service response, if the CM-air-ASE is in the *CONTACT* state the CM-air-ASE shall:

a) create a CMAircraftMessage APDU with cmContactResponse APDU-element based on the CM-contact *Contact Response* parameter value,

b) invoke D-START response with the following:

1) the abstract value “rejected (permanent)” as D-START *Result* parameter value, and

2) the CMAircraftMessage APDU as the D-START *User Data* parameter value, and

c) enter the *IDLE* state.

2.1.5.3.2.7.2 Upon receipt of a CM-contact service response, if the CM-air-ASE is in the *CONTACT DIALOGUE* state the CM-air-ASE shall:

a) create a CMAircraftMessage APDU with a cmContactResponse APDU-element based on the CM-contact *Contact Response* parameter value,

b) invoke D-DATA request with the CMAircraftMessage APDU as the D-DATA *User Data* parameter value, and

c) enter the *DIALOGUE* state.

2.1.5.3.2.8 CM-user-abort Service Request

2.1.5.3.2.8.1 Upon receipt of a CM-user-abort service request, if the CM-air-ASE is not in the *IDLE* state the CM-air-ASE shall:

a) stop timer t_{logon}, if set,

b) invoke D-ABORT request with the D-ABORT *Originator* parameter set to the abstract value “user”, and

c) enter the *IDLE* state.

2.1.5.3.2.9 D-ABORT Indication

2.1.5.3.2.9.1 Upon receipt of a D-ABORT indication, if the CM-air-ASE is not in the *IDLE* state the CM-air-ASE shall:

a) stop timer t_{logon}, if set

b) if the CM-air-user is an active user, then:

1) if the D-ABORT *Originator* parameter contains the abstract value "user" invoke CM-user-abort service indication, or

2) else invoke CM-provider-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CM-provider-abort service *Reason* parameter value, and

c) enter *IDLE* state.

2.1.5.3.2.10 D-P-ABORT Indication

2.1.5.3.2.10.1 Upon receipt of a D-P-ABORT indication, if the CM-air-ASE is not in the *IDLE* state the CM-air-ASE shall:

a) stop timer t_{logon}, if set,

b) if the CM-air-user is an active user, invoke CM-provider-abort service indication with the CM-provider-abort *Reason* parameter set to the abstract value "communication-service-failure", and

c) enter the *IDLE* state.

2.1.5.3.3 CM-Ground-ASE Protocol Description

Note 1.— The states defined for the CM-ground-ASE are the following:

a) IDLE,

b) LOGON,

c) UPDATE,

d) CONTACT,

e) DIALOGUE,

f) CONTACT DIALOGUE,

g) END, and

h) FORWARD.

Note 2.— The CM-ground-user is considered an active user from the time:

a) the CM-ground-user receives a CM-logon Ind until it:

1) invokes a CM-logon Rsp, if a dialogue is not maintained,

2) invokes a CM-end Req, if a dialogue is maintained,

3) receives a CM-user abort,

4) receives a CM-provider abort, or

5) invokes a CM-user abort,

b) the CM-ground-user invokes a CM-contact Req until it

1) receives a CM-contact Cnf, if a dialogue is not maintained,

2) receives a CM-user abort,

3) receives a CM-provider abort, or

4) invokes a CM-user abort

c) the CM-ground-user invokes a CM-forward Req until it

1) receives a CM-forward Cnf,

2) receives a CM-provider abort, or

3) invokes a CM-user abort.

2.1.5.3.3.1 On initiation, the CM-ground-ASE shall be in the *IDLE* state.

2.1.5.3.3.2 D-START Indication

2.1.5.3.3.2.1 Upon receipt of a D-START indication, if the CM-ground-ASE is in the *IDLE* state and the APDU contained in the D-START *User Data* parameter is a [CMLogonRequest] APDU and the abstract value of the D-START *Calling Peer ID* parameter is a 24 bit Aircraft Address and the D-START indication *Priority* Quality-of-Service parameter has the abstract value "flight regularity communications" and the *RER* Quality-of-Service parameter has the abstract value of "low", then:

2.1.5.3.3.2.1.1 If the D-START *DS User Version Number* parameter value is greater than the CM-ground-ASE version number the CM-ground-ASE shall:

a) invoke D-START response with the following:

1) the CM-ground-ASE version number as the D-START *DS User Version Number* parameter value, and

2) the abstract value of "rejected (permanent)" as the D-START *Result* parameter value, and

b) enter the *IDLE* state.

2.1.5.3.3.2.1.2 If the D-START *DS User Version Number* parameter value is less than the CM-ground-ASE version number the CM-ground-ASE shall:

a) invoke CM-logon service indication with the following:

1) the D-START *Calling Peer ID* parameter value as the CM-logon service *Aircraft Address* parameter value,

2) the D-START *DS User Version Number* parameter value as the CM-logon service *CM ASE Version Number* parameter value, and

3) the APDU in the D-START *User Data* parameter as the CM-logon service *Logon Request* parameter value, and

b) enter the *LOGON* state.

2.1.5.3.3.2.1.3 If the D-START *DS User Version Number* parameter value is equal to CM-ground-ASE version number the CM-ground-ASE shall:

a) invoke CM-logon service indication with:

1) the D-START *Calling Peer ID* parameter value as the CM-logon service *Aircraft Address* parameter value, and

2) the APDU in the D-START *User Data* parameter as the CM-logon service *Logon Request* parameter value, and

b) enter the *LOGON* state

2.1.5.3.3.2.2 Upon receipt of a D-START indication, if the receiving CM-ground-ASE is in the *IDLE* state and the APDU contained in the D-START *User Data* parameter is a [CMForwardRequest] APDU and the abstract value of the D-START *Calling Peer ID* parameter is a 4 to 8 character Facility Designation and the D-START indication *Priority* Quality-of-Service parameter has the abstract value "flight regularity communications" and the *RER* Quality-of-Service parameter has the abstract value of "low", then:

2.1.5.3.3.2.2.1 If the CM-ground-ASE does not support the CM-forward service, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmForwardResponse [service-not-supported] APDU message element,

b) invoke D-START response with:

1) the APDU as the D-START *User Data* parameter value, and

2) the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and

c) remain in the *IDLE* state.

2.1.5.3.3.2.2.2 If the D-START *DS User Version Number* parameter value is greater than the CM-ground-ASE version number and the CM-ground-ASE supports the CM-forward service, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmForwardResponse [incompatible-version] APDU message element,

b) invoke D-START response with the following:

1) the CM-ground-ASE version number as the D-START *DS User Version Number* parameter value, and

2) the APDU as the D-START *User Data* parameter value,

3) the abstract value of "rejected (permanent)" as the D-START *Result* parameter value, and

c) remain in the *IDLE* state.

2.1.5.3.3.2.2.3 If the D-START *DS User Version Number* parameter value is less than the CM-ground-ASE version number and the CM-ground-ASE supports the CM-forward service, the CM-ground-ASE shall:

a) invoke CM-forward service indication with the following:

1) the D-START *Calling Peer ID* parameter value as the CM-forward service *Calling Facility Designation* parameter value,

2) the D-START *DS User Version Number* parameter value as the CM-forward service *CM ASE Version Number* parameter value, and

3) the APDU in the D-START *User Data* parameter as the CM-forward service *Forward Request* parameter value,

b) create a CMGroundMessage APDU with a cmForwardResponse [success] APDU message element,

c) invoke D-START response with the following:

1) the APDU as the D-START *User Data* parameter value, and

2) the abstract value of "rejected (permanent)" as the D-START *Result* parameter value, and

d) remain in the *IDLE* state.

Note.— This assumes that CM ASEs are backwards compatible.

2.1.5.3.3.2.2.4 If the D-START *DS User Version Number* parameter value is equal to the CM-ground-ASE version number and the CM-ground-ASE supports the CM-forward service, the CM-ground-ASE shall:

a) invoke CM-forward service indication with the following:

1) the D-START *Calling Peer ID* parameter value as the CM-forward service *Calling Facility Designation* parameter value, and

2) the APDU in the D-START *User Data* parameter as the CM-forward service *Forward Request* parameter value,

b) create a CMGroundMessage APDU with a cmForwardResponse [success] APDU message element,

c) invoke D-START response with the following:

1) the APDU as the D-START *User Data* parameter value, and

2) the abstract value of "rejected (permanent)" as the D-START *Result* parameter value, and

d) remain in the *IDLE* state.

2.1.5.3.3.3 D-START Confirmation

2.1.5.3.3.3.1 Upon receipt of a D-START confirmation:

2.1.5.3.3.3.1.1 If the CM-ground-ASE is in the *UPDATE* state and the D-START *User Data* parameter is not provided, the CM-ground-ASE shall:

a) stop timer t_{update}, and

b) if the abstract value of the D-START *Result* parameter is "rejected (permanent)" and the abstract value of the D-START *Reject Source* parameter is "DS user", enter the *IDLE* state.

2.1.5.3.3.3.1.2 If the CM-ground-ASE is in the *CONTACT* state and the APDU contained in the D-START *User Data* parameter is a [CMContactResponse] APDU, the CM-ground-ASE shall:

a) stop timer $t_{contact}$, and

b) if the abstract value of the D-START *Result* parameter is "rejected (permanent)" and the abstract value of the D-START *Reject Source* parameter is "DS user" then:

1) invoke CM-contact service confirmation with the APDU in the D-START *User Data* parameter as the CM-contact *Contact Response* parameter value, and

2) enter the *IDLE* state.

2.1.5.3.3.3.1.3 If the CM-ground-ASE is in the *FORWARD* state and if the D-START *User Data* parameter is a [CMForwardResponse] APDU, and the abstract value of the D-START *Result* parameter is "rejected (permanent)" and the abstract value of the D-START *Reject Source* parameter is "DS user", the CM-ground-ASE shall:

a) stop timer $t_{forward}$,

b) if the abstract value of the D-START *User Data* parameter is "service-not-supported", the CM-ground-ASE will invoke a CM-forward service confirmation with the CM-forward *Result* parameter set to the abstract value "service-not-supported",

c) if the abstract value of the D-START *User Data* parameter is "incompatible-version", the CM-ground-ASE will invoke a CM-forward service

confirmation with the *DS User Version Number* parameter value as the CM-forward *CM ASE Version Number* parameter and set the CM-forward *Result* parameter abstract value to "incompatible-version",

d) if the abstract value of the D-START *User Data* parameter is "success", the CM-ground-ASE will invoke a CM-forward service confirmation with the CM-forward *Result* parameter set to the abstract value "success", and

e) enter the *IDLE* state.

2.1.5.3.3.4 D-DATA Indication

2.1.5.3.3.4.1 Upon receipt of a D-DATA indication if the CM-ground-ASE is in the *CONTACT DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a [CMContactResponse] APDU, the CM-ground-ASE shall:

a) invoke CM-contact service confirmation with the APDU contained in the D-DATA *User Data* parameter as the CM-contact *Contact Response* parameter value, and

b) enter the *DIALOGUE* state.

2.1.5.3.3.5 D-END Confirmation

2.1.5.3.3.5.1 Upon receipt of a D-END confirmation, if the CM-ground-ASE is in the *END* state and the abstract value of the D-END *Result* is "accepted", the CM-ground-ASE shall:

a) stop timer t_{end}, and

b) enter the *IDLE* state.

2.1.5.3.3.6 CM-logon Service Response

2.1.5.3.3.6.1 Upon receipt of a CM-logon service response, if the CM-ground-ASE is in the *LOGON* state the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmLogonResponse APDU-element based on the CM-logon *Logon Response* parameter value, and

b) invoke D-START response with the following:

1) the CMGroundMessage APDU as the D-START *User Data* parameter value,

2) if the CM-logon *Maintain Dialogue* parameter is provided by the CM-ground-user:

i) set the abstract value "accepted" as the D-START *Result* parameter value, and

ii) enter the *DIALOGUE* state.

3) if the CM-logon *Maintain Dialogue* parameter is not provided by the CM-ground-user:

i) set the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and

ii) enter the *IDLE* state.

2.1.5.3.3.7 CM-update Service Request

2.1.5.3.3.7.1 Upon receipt of a CM-update service request, if the CM-ground-ASE is in the *IDLE* state, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmUpdate APDU-element based on the CM-update *Update Information* parameter value,

b) invoke D-START request with the following:

1) the CM-update *Aircraft Address* parameter value as the D-START *Called Peer ID* parameter value,

2) the CM-update *Facility Designation* parameter value as the D-START *Calling Peer ID* parameter value,

3) set the D-START *Quality of Service* parameter as follows:

i) if provided, the CM-update service *Class of Communication Service* parameter value as the D-START *Routing Class* parameter value,

ii) the abstract value of "flight regularity communications", as the D-START *Priority* parameter value, and

iii) the abstract value of "low" as the D-START *RER* parameter value,

4) the CMGroundMessage APDU as the D-START *User Data* parameter value,

c) start timer t_{update}, and

d) enter the *UPDATE* state.

2.1.5.3.3.7.2 Upon receipt of a CM-update service request, if the CM-ground-ASE is in the *DIALOGUE* state, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmUpdate APDU-element based on the CM-update *Update Information* parameter value,

b) invoke D-DATA request with the CMGroundMessage APDU as the D-DATA *User Data* parameter value, and

c) remain in the *DIALOGUE* state.

2.1.5.3.3.8 CM-contact Service Request

2.1.5.3.3.8.1 Upon receipt of a CM-contact service request, if the CM-ground-ASE is in the *IDLE* state the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmContactRequest APDU-element based on the CM-contact *Contact Request* parameter value,

b) invoke D-START request with the following:

1) the CM-contact *Aircraft Address* parameter value as the D-START *Called Peer ID* parameter value,

2) the CM-contact *Facility Designation* parameter value as the D-START *Calling Peer ID* parameter value,

3) set the D-START *Quality of Service* parameters as follows:

i) if provided, the CM-contact service *Class of Communication Service* parameter value as the D-START *Routing Class* parameter value,

ii) The abstract value of “flight regularity communications”, as the D-START *Priority* parameter value, and

iii) The abstract value of “low” as the D-START *RER* parameter value,

4) the CMGroundMessage APDU as the D-START *User Data* parameter value,

c) start timer $t_{contact}$, and

d) enter the *CONTACT* state.

2.1.5.3.3.8.2 Upon receipt of a CM-contact service request, if the CM-ground-ASE is in the *DIALOGUE* state the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmContactRequest APDU-element based on the CM-contact *Contact Request* parameter value,

b) invoke D-DATA request with the CMGroundMessage APDU as the D-DATA *User Data* parameter value, and

c) enter the *CONTACT DIALOGUE* state.

2.1.5.3.3.9 CM-end Service Request

2.1.5.3.3.9.1 Upon receipt of a CM-end service request, if the CM-ground-ASE is in the *DIALOGUE* state the CM-ground-ASE shall:

a) invoke D-END request,

b) start timer t_{end} , and

c) enter the *END* state.

2.1.5.3.3.10 CM-forward Service Request

2.1.5.3.3.10.1 Upon receipt of a CM-forward service request, if the CM-ground-ASE is in the *IDLE* state, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmForwardRequest APDU-element based on the CM-forward service *Forward Request* parameter value,

b) invoke D-START request with the following:

1) the CM-forward *Called Facility Designation* parameter value as the D-START *Called Peer ID* parameter value,

2) the CM-forward *Calling Facility Designation* parameter value as the D-START *Calling Peer ID* parameter value,

3) the sending CM-ground-ASE version number as the D-START *DS User Version Number* parameter value,

4) set the D-START *Quality of Service* parameter as follows:

i) if provided, the CM-forward service *Class of Communication Service* parameter value as the D-START *Routing Class* parameter value,

ii) the abstract value of "flight regularity communications" as the D-START *Priority* parameter value, and

iii) the abstract value of "low" as the D-START *RER* parameter value,

5) the CMGroundMessage APDU as the D-START *User Data* parameter value,

c) start timer $t_{forward}$, and

d) enter the *FORWARD* state.

2.1.5.3.3.11 CM-user-abort Service Request

2.1.5.3.3.11.1 Upon receipt of a CM-user-abort service request, if the CM-ground-ASE is not in the *IDLE* state the CM-ground-ASE shall:

a) stop any timer, if set,

b) invoke D-ABORT request with the D-ABORT *Originator* parameter set to the abstract value "user", and

c) enter the *IDLE* state.

2.1.5.3.3.12 D-ABORT Indication

2.1.5.3.3.12.1 Upon receipt of a D-ABORT indication, if the CM-ground-ASE is not in the *IDLE* state the CM-ground-ASE shall:

a) stop any timer, if set,

b) if the CM-ground-user is an active user, then:

1) if the D-ABORT *Originator* parameter contains the abstract value "user" invoke CM-user-abort service indication,

2) else invoke CM-provider-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CM-provider-abort service *Reason* parameter value, and

c) enter *IDLE* state.

2.1.5.3.3.13 D-P-ABORT Indication

2.1.5.3.3.13.1 Upon receipt of a D-P-ABORT indication, if the CM-ground-ASE is not in the *IDLE* state the CM-ground-ASE shall:

a) stop any timer, if set,

b) if the CM-ground-user is an active user, invoke CM-provider-abort service indication with the CM-provider-abort *Reason* parameter set to the abstract value "communication-service-failure", and

c) enter the *IDLE* state.

2.1.5.4 Exception Handling

2.1.5.4.1 Timer Expiration

2.1.5.4.1.1 If a CM-ASE detects that a timer has expired, that CM-ASE shall:

a) stop all timers,

b) interrupt any current activity,

c) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [timer-expired] APDU message element,

d) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [timer-expired] APDU message element,

e) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value,

f) if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value "timer-expired" as the CM-provider-abort *Reason* parameter value, and

g) enter the *IDLE* state.

2.1.5.4.2 Unrecoverable System Error

2.1.5.4.2.1 **Recommendation.—** *If a CM-ASE has an unrecoverable system error, the CM-ASE should:*

a) *stop all timers,*

b) *if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [undefined-error] APDU message element,*

c) *if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [undefined-error] APDU message element,*

d) *invoke D-ABORT request with:*

1) *the abstract value "provider" as the D-ABORT* Originator *parameter value, and*

2) *the APDU as the D-ABORT* User Data *parameter value,*

e) *if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value "undefined-error" as the CM-provider-abort* Reason *parameter value, and*

f) *enter the* IDLE *state.*

2.1.5.4.3 Invalid PDU

2.1.5.4.3.1 If the *User Data* parameter of a D-START indication or D-DATA indication does not contain a valid PDU as defined in 2.1.5.3.1.3 and 2.1.5.3.1.4, the CM-ASE shall:

a) stop all timers,

b) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [invalid-PDU] APDU message element,

c) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [invalid-PDU] APDU message element,

d) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value,

e) if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value “invalid-PDU” as the CM-provider-abort *Reason* parameter value, and

f) enter the *IDLE* state.

2.1.5.4.3.2 If the *User Data* parameter of a D-START confirmation does not contain a valid PDU as defined in 2.1.5.3.1.3 and 2.1.5.3.1.4, the CM-ASE shall:

a) stop all timers,

b) if the D-START *Result* parameter is set to the abstract value “accepted”, then

1) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [invalid-PDU] APDU message element,

2) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [invalid-PDU] APDU message element,

3) invoke D-ABORT request with:

i) the abstract value “provider” as the D-ABORT *Originator* parameter value, and

ii) the APDU as the D-ABORT *User Data* parameter value,

c) if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value “invalid-PDU” as the CM-provider-abort *Reason* parameter value, and

d) enter the *IDLE* state.

2.1.5.4.4 Not Permitted PDU

2.1.5.4.4.1 If the *User Data* parameter of a D-START indication or D-DATA indication is a valid PDU, but is not a permitted PDU as defined within 2.1.5.3.1.2, the CM-ASE shall:

a) stop all timers,

b) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [not-permitted-PDU] APDU message element,

c) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [not-permitted-PDU] APDU message element,

d) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value,

e) if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value "not-permitted-PDU" as the CM-provider-abort *Reason* parameter value, and

f) enter the *IDLE* state.

2.1.5.4.4.2 If the *User Data* parameter of a D-START confirmation is a valid PDU, but is not a permitted PDU as defined within 2.1.5.3.1.2, the CM-ASE shall:

a) stop all timers,

b) if the D-START *Result* parameter is set to the abstract value "accepted":

1) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [not-permitted-PDU] APDU message element,

2) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [not-permitted-PDU] APDU message element,

3) invoke D-ABORT request with:

i) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

ii) the APDU as the D-ABORT *User Data* parameter value,

c) if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value "not-permitted-PDU" as the CM-provider-abort *Reason* parameter value, and

d) enter the *IDLE* state.

2.1.5.4.5 D-START Confirmation *Result* or *Reject Source* Parameter Values Not as Expected

2.1.5.4.5.1 If the CM-ground-ASE receives a D-START confirmation with the D-START *Result* parameter having the abstract value of "accepted", the CM-ground-ASE shall:

a) stop all timers,

b) create a CMGroundMessage APDU with a cmAbortReason [dialogue-acceptance-not-permitted] APDU message element,

c) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value,

d) if the CM-ground-user is an active user, invoke CM-provider-abort service indication with the abstract value "dialogue-acceptance-not-permitted" as the CM-provider-abort *Reason* parameter value, and

e) enter the *IDLE* state.

2.1.5.4.5.2 If the CM-ASE receives a D-START confirmation with the D-START *Result* parameter having the abstract value of "rejected (transient)" or if the D-START *Reject Source* parameter has the abstract value of "DS provider", and if the CM-user is an active user, the CM-ASE shall invoke CM-provider-abort service indication with the abstract value "communication-service-error" APDU as the CM-provider-abort *Reason* parameter value.

2.1.5.4.6 D-END Confirmation Not as Expected

2.1.5.4.6.1 If the CM-ground-ASE receives a D-END confirmation with the D-END *Result* parameter that does not have the abstract value of "accepted", the CM-ground-ASE shall:

a) stop all timers,

b) create a CMGroundMessage APDU with a cmAbortReason [dialogue-end-not-accepted] APDU message element,

c) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) enter the *IDLE* state.

2.1.5.4.7 D-START Indication *Quality of Service* Parameter Not as Expected

2.1.5.4.7.1 If the abstract value of the *Priority* Quality-of-Service parameter is not "flight regularity communications" or the abstract value of the *RER* Quality-of-Service parameter is not "low", the CM-ASE shall:

a) stop all timers

b) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a CMAbortReason [invalid-QOS-parameter] APDU message element,

c) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a CMAbortReason [invalid-QOS-parameter] APDU message element,

d) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

e) enter the *IDLE* state

2.1.5.4.8 Expected PDU Not Provided

2.1.5.4.8.1 If the *User Data* parameter of a D-START indication, D-START confirmation (with the *Result* parameter set to the abstract value "accepted"), or D-DATA indication is not provided where it is expected, the CM-ASE shall:

a) stop all timers,

b) if the CM-ASE is a CM-air-ASE, create a CMAircraftMessage APDU with a cmAbortReason [expected-PDU-missing] APDU message element,

c) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [expected-PDU-missing] APDU message element,

d) invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value,

e) invoke a CM-provider-abort service indication with the abstract value "expected-PDU-missing", and

f) enter the *IDLE* state.

2.1.5.4.8.2 If the *User Data* parameter of a D-START confirmation (with the *Result* parameter set to the abstract value "rejected (transient)" or "rejected (permanent)") is not provided where it is expected, the CM-ASE shall:

a) stop all timers,

b) invoke a CM-provider-abort service indication with the abstract value "expected-PDU-missing", and

c) enter the *IDLE* state.

2.1.5.5 CM ASE State Tables

2.1.5.5.1 Priority

2.1.5.5.1.1 If the state tables for the CM-air-ASE and the CM-ground-ASE shown below conflict with textual statements made elsewhere in this document, the textual statements shall take precedence.

Note 1.— In the following state tables, the statement "cannot occur" means that if the implementation conforms to the SARPs, it is impossible for this event to occur. If the event does occur, this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE aborts with the error "unrecoverable system error".

Note 2.— In the following state tables, the statement "not permitted" means that the implementation must prevent this event from occurring through some local means. If the event does occur this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE performs a local rejection of the request rather than aborting the dialogue.

Table 2.1.5-2. CM-Ground-ASE State Table

STATE ⇒ EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*
DIALOGUE Service Events								
D-START Indication *Version Number* is greater than the CM-ground-ASE *version number*, ground-ground forwarding supported	**!** D-START response →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication *Version Number* is less than or equal to the CM-ground-ASE version number, *User Data* = CMLogonRequest, ground-ground forwarding supported	**!** CM-logon indication →*LOGON*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication *Version Number* is less than or equal to the CM-ground-ASE version number, *User Data* = CMForwardRequest, ground-ground forwarding supported	**!** CM-forward indication **!** D-START response →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication, Ground-ground forwarding is not supported, *User Data* = CMForwardRequest	**!** D-START response →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user", D-START *User Data* parameter not provided	cannot occur	cannot occur	**!** Stop timer t_{update} →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user", D-START User *Data*=CMContactResponse	cannot occur	cannot occur	cannot occur	**!** Stop timer $t_{contact}$ **!** CM-contact confirmation →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur

STATE ⇒ EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user", D-START User *Data*=CMForwardResponse	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	**!** Stop timer $t_{forward}$ **!** CM-forward confirmation ⇒*IDLE*
D-DATA Indication	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	**!** CM contact confirmation ⇒*DIALOGUE*	cannot occur	cannot occur
D-END Confirmation *Result* "accepted"	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	**!** stop timer t_{end} ⇒*IDLE*	cannot occur
CM-User Events								
CM-update Request	**!** D-START request **!** Start timer t_{update} ⇒*UPDATE*	not permitted	not permitted	not permitted	**!** D-DATA request ⇒*DIALOGUE*	not permitted	not permitted	not permitted
CM-contact Request	**!** D-START request **!** Start timer $t_{contact}$ ⇒*CONTACT*	not permitted	not permitted	not permitted	**!** D-DATA request ⇒*CONTACT DIALOGUE*	not permitted	not permitted	not permitted
CM-forward Request	**!** D-START request **!** Start timer $t_{forward}$ ⇒*FORWARD*	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted
CM-logon Response *Maintain Dialogue* not supplied by CM-ground user	not permitted	**!** D-START response ⇒*IDLE*	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted
CM-logon Response *Maintain Dialogue* "accepted"	not permitted	**!** D-START response ⇒*DIALOGUE*	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted
CM-end Request	not permitted	not permitted	not permitted	not permitted	**!** D-END request **!** Start timer t_{end} ⇒*END*	not permitted	not permitted	not permitted
ABORT Events								

STATE ⇒ EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*
CM-user-abort Request	not permitted	**!** D-ABORT request ⇒*IDLE*	**!** stop timer t_{update} **!** D-ABORT request ⇒*IDLE*	**!** stop timer $t_{contact}$ **!** D-ABORT request ⇒*IDLE*	**!** D-ABORT request ⇒*IDLE*	**!** D-ABORT request ⇒*IDLE*	not permitted	**!** stop timer $t_{forward}$ **!** D-ABORT request ⇒*IDLE*
D-ABORT Indication *Originator* is "provider"	cannot occur	**!** CM-provider-abort indication ⇒*IDLE*	**!** stop timer t_{update} **!** CM-provider-abort indication ⇒*IDLE*	**!** stop timer $t_{contact}$ **!** CM-provider-abort indication ⇒*IDLE*	**!** CM-provider-abort indication ⇒*IDLE*	**!** CM-provider-abort indication ⇒*IDLE*	**!** stop timer t_{end} ⇒*IDLE*	**!** stop timer $t_{forward}$ **!** CM-provider-abort indication ⇒*IDLE*
D-ABORT Indication *Originator* is "user"	cannot occur	**!** CM-user-abort indication ⇒*IDLE*	**!** stop timer t_{update} **!** CM-user-abort indication ⇒*IDLE*	**!** stop timer $t_{contact}$ **!** CM-user-abort indication ⇒*IDLE*	**!** CM-user-abort indication ⇒*IDLE*	**!** CM-user-abort indication ⇒*IDLE*	**!** stop timer t_{end} ⇒*IDLE*	cannot occur
D-P-ABORT indication	cannot occur	**!** CM-provider-abort indication ⇒*IDLE*	**!** stop timer t_{update} **!** CM-provider-abort indication ⇒*IDLE*	**!** stop timer $t_{contact}$ **!** CM-provider-abort indication ⇒*IDLE*	**!** CM-provider-abort indication ⇒*IDLE*	**!** CM-provider-abort indication ⇒*IDLE*	**!** stop timer t_{end} ⇒*IDLE*	**!** stop timer $t_{forward}$ **!** CM-provider-abort indication ⇒*IDLE*
T_{update} Expires	cannot occur	cannot occur	**!** D-ABORT request **!** CM-provider-abort indication ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
$T_{contact}$ Expires	cannot occur	cannot occur	cannot occur	**!** D-ABORT request **!** CM-provider-abort indication ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur

STATE ⇒ EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*
T_{end} Expires	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	**!** D-ABORT request ⇒*IDLE*	cannot occur
$T_{forward}$ Expires	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	**!** D-ABORT request **!** CM-provider-abort indication ⇒*IDLE*

Table 2.1.5-3. CM-Air-ASE State Table

STATE ⇒ EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*
DIALOGUE Service Events					
D-START Indication *User Data* CMUpdate	**!** CM-update indication **!** D-START response ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication *User Data* CMContactRequest	**!** CM-contact indication ⇒*CONTACT*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user"	cannot occur	**!** Stop timer t_{logon} **!** CM-logon confirmation ⇒*IDLE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted"	cannot occur	**!** Stop timer t_{logon} **!** CM-logon confirmation ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur
D-DATA Indication *User Data* CMUpdate	cannot occur	cannot occur	cannot occur	**!** CM-update indication ⇒*DIALOGUE*	cannot occur

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*
D-DATA Indication *User Data* CMContactRequest	cannot occur	cannot occur	cannot occur	**!** CM-contact indication →*CONTACT DIALOGUE*	cannot occur
D-END Indication	cannot occur	cannot occur	cannot occur	**!** CM-end indication **!** D-END response →*IDLE*	cannot occur
CM-User Events					
CM-contact Response	not permitted	not permitted	**!** D-START response →*IDLE*	not permitted	**!** D-DATA request →*DIALOGUE*
CM-logon Request	**!** D-START request **!** Start timer t_{logon} →*LOGON*	not permitted	not permitted	not permitted	not permitted
ABORT Events					
CM-user-abort Request	not permitted	**!** stop timer t_{logon} **!** D-ABORT request →*IDLE*	**!** D-ABORT request →*IDLE*	**!** D-ABORT request →*IDLE*	**!** D-ABORT request →*IDLE*
D- ABORT Indication *Originator* is “provider”	cannot occur	**!** stop timer t_{logon} **!** CM-provider-abort indication →*IDLE*	**!** CM-provider-abort indication →*IDLE*	**!** CM-provider-abort indication →*IDLE*	**!** CM-provider-abort indication →*IDLE*
D-ABORT Indication *Originator* is “user”	cannot occur	**!** stop timer t_{logon} **!** CM-user-abort indication →*IDLE*	**!** CM-user-abort indication →*IDLE*	**!** CM-user-abort indication →*IDLE*	**!** CM-user-abort indication →*IDLE*

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*
D-P-ABORT indication	cannot occur	**!** stop timer t_{logon} **!** CM-provider-abort indication →*IDLE*	**!** CM-provider-abort indication →*IDLE*	**!** CM-provider-abort indication →*IDLE*	**!** CM-provider-abort indication →*IDLE*
T_{logon} Expires	cannot occur	**!** D-ABORT request **!** CM-provider-abort indication →*IDLE*	cannot occur	cannot occur	cannot occur

2.1.6 COMMUNICATION REQUIREMENTS

2.1.6.1 Encoding Rules

2.1.6.1.1 The CM application shall use PER encoding as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in 2.1.4.

2.1.6.2 Dialogue Service Requirements

2.1.6.2.1 Primitive Requirements

2.1.6.2.1.1 Where dialogue service primitives, that is D-START, D-DATA, D-END, D-ABORT, and D-P-ABORT are described as being invoked in 2.1.5, the CM-ground-ASE and the CM-air-ASE shall exhibit external behaviour consistent with the dialogue service, as described in 4.2, having been implemented and its primitives invoked.

2.1.6.2.2 ATN Quality-of-Service Requirements

2.1.6.2.2.1 The *Priority* Quality-of-Service parameter of the D-START for CM shall be the abstract value of “flight regularity communications”.

2.1.6.2.2.2 The *RER* Quality-of-Service parameter of the D-START for CM shall be set to the abstract value of “low”.

2.1.6.2.2.3 The CM-ASE shall map the Class of Communication Service abstract values to the Routing Class abstract value part of the D-START Quality-of-Service parameter as presented in Table 2.1.6-1.

Table 2.1.6-1. Mapping Between Class of Communication and Routing Class Abstract Values

Class of Communication Abstract Value	Routing Class Abstract Value
A	Traffic follows Class A ATSC route(s)
B	Traffic follows Class B ATSC route(s)
C	Traffic follows Class C ATSC route(s)
D	Traffic follows Class D ATSC route(s)
E	Traffic follows Class E ATSC route(s)
F	Traffic follows Class F ATSC route(s)
G	Traffic follows Class G ATSC route(s)
H	Traffic follows Class H ATSC route(s)

Note.— ATSC values are defined in 1.3.

2.1.7 CM USER REQUIREMENTS

Note.— Requirements imposed on CM-users concerning CM messages and interfacing with the CM-ASEs are presented in 2.1.7.

2.1.7.1 CM-Air-User Requirements

Note 1.— When a CM-air-user invokes the CM-logon service and requires a particular class of communication service, it sets the Class of Communication Service *parameter to be the class of communication it requires.*

Note 2.— When the CM-air-user invokes a CM-logon service and has no preference for the class of communication service to be used, the Class of Communication Service *parameter does not need to be provided.*

Note 3.— For each CM-air-ASE invocation, the CM-air-user establishes a correlation between a CM-air-ASE invocation and the facility designation.

Note 4.— Upon the initiation of a CM-logon service request, or upon receipt of a CM-update service indication or a CM-contact service indication, the ASE invocation correlation is based on the facility designation in the Facility Designation *parameter of the respective CM service.*

Note 5.— The correlation is maintained for the duration of the ASE invocation.

2.1.7.1.1 CM-logon Service Requirements

Note.— Only the CM-air-user is permitted to initiate the CM-logon service.

2.1.7.1.1.1 When invoking the CM-logon service request the CM-air-user shall provide the following as part of the CMLogonRequest:

a) its CM long TSAP,

b) the aircraft's Flight ID,

c) information on each application for which it requires a data link service as follows:

1) for air-only initiated services: application name and version number for all the versions that can be supported, and

2) for applications that can be ground initiated: application name, version number, and address for all the versions that can be supported, and

d) flight information data as required by the ground system.

2.1.7.1.1.2 When invoking the CM-logon service request, if any RDP for a given application address is different than the CM RDP, the CM-air-user shall use the long TSAP for each application address provided.

Note.— The long TSAP = RDP + short TSAP. The short TSAP = ARS + LOC + SYS + NSEL + TSEL. The RDP = VER + ADM + RDF.

2.1.7.1.1.3 When invoking the CM-logon service request, the ARS component of the short TSAP shall contain the aircraft's 24 bit address.

Note.— If there is more than one routing domain on the aircraft, the LOC field is used to differentiate them.

2.1.7.1.1.4 Upon receipt of a CM-logon service confirmation, the CM-air-user shall create the actual TSAP for each ground application information contained in the *Logon Response* based on the IDP and long TSAP for each application as defined in 2.1.4.

Note.— The actual TSAP = IDP + long TSAP. The IDP = AFI + IDI.

2.1.7.1.1.5 Upon receipt of a CM-logon service confirmation, the CM-air-user shall make the information contained in the CMLogonResponse available to the other applications (i.e., ADS, CPDLC, and FIS), as well as to the dialogue service provider.

2.1.7.1.1.6 Upon the receipt of a *Logon Response* from a CM-logon service confirmation from a ground facility for which CM information has previously been received, the CM-air-user shall only replace the previous information for which new logon information has been received.

2.1.7.1.2 CM-update Service Requirements

2.1.7.1.2.1 Upon the receipt of *Update Information* from a CM-update service indication from a ground facility designation for which CM information has previously been received, the CM-air-user shall only replace the previous information for which updated information has been received.

2.1.7.1.2.2 Upon receipt of a CM-update service indication, the CM-air-user shall create the actual TSAP for each ground application information contained in the *Update Information* based on the IDP and long TSAP for each application as defined in 2.1.4.

Note.— The actual TSAP = IDP + long TSAP. The IDP = AFI + IDI.

2.1.7.1.2.3 The CM-air-user shall make the updated information contained in the *Update Information* available to the other applications (i.e., ADS, CPDLC, and FIS), as well as to the dialogue service provider.

2.1.7.1.3 CM-contact Service Requirements

2.1.7.1.3.1 **Recommendation.** — *Upon receipt of a CM-contact indication, the CM-air-user should invoke the CM-logon request with the indicated ground system within 0.5 seconds.*

2.1.7.1.3.2 Upon receipt of a CM-logon confirmation when performing the CM-contact service, the CM-air-user shall invoke a CM-contact response.

2.1.7.1.3.3 **Recommendation.** — *Upon receipt of a CM-logon confirmation when performing the CM-contact service, the CM-air-user should invoke a CM-contact response within 0.5 seconds.*

2.1.7.1.3.4 Upon receipt of a CM-contact service indication, the CM-air-user shall attempt to initiate a CM-logon service request with the indicated ground system.

Note.— If a CM-logon service request is initiated, the CM-air-user will comply with the CM-logon requirements as stated in 2.1.7.1.1.

2.1.7.1.3.5 In addition to the above CM-logon service requirements, upon receipt of a CM-logon service response from the indicated facility designation, or if no CM-logon service request can be initiated, the CM-air-user shall invoke the CM-contact service response indicating the success or lack thereof of the CM-logon service request.

2.1.7.2 CM-Ground-User Requirements

2.1.7.2.1 General CM-Ground-User Requirements.

2.1.7.2.1.1 A CM-ground-user shall invoke the CM-logon service, CM-update service, CM-contact service, and CM-end service only when communicating with a CM-air-user.

2.1.7.2.1.2 A CM-ground-user shall invoke the CM-forward service only when communicating with another CM-ground-user.

Note 1.— When a CM-ground-user invokes the CM-update service, CM-contact service, or CM-forward service and requires a particular class of communication service, it will set the Class of Communication Service *parameter to be the class of communication it requires.*

Note 2.— When the CM-ground-user invokes a CM-update service, CM-contact service, or CM-forward service and has no preference for the class of communication service to be used, the Class of Communication Service *parameter does not need to be provided.*

Note 3.— When a CM-ground-user specifies the Class of Communication Service *parameter and the dialogue is in place, the class of communication parameter is ignored.*

Note 4.— For each CM-ground-ASE invocation, the CM-ground-user establishes a correlation between a CM-ground-ASE invocation and the aircraft 24 bit address

Note 5.— Upon the initiation a CM-update service request or CM-contact service request, or upon receipt of a CM-logon service indication the ASE invocation correlation is based on the 24-bit aircraft address in the Aircraft Address *parameter of the respective CM service.*

Note 6.— The correlation is maintained for the duration of the ASE invocation.

2.1.7.2.2 CM-logon Service Requirements

2.1.7.2.2.1 **Recommendation.** — *Upon receipt of a CM-logon indication, the CM-ground-user should invoke the CM-logon response within 0.5 seconds.*

2.1.7.2.2.2 Upon receipt of a CM-logon service indication, the CM-ground-user shall make the aircraft application information contained in the *Logon Request* available to the other applications (i.e., ADS, CPDLC, and FIS), as well as to the dialogue service provider.

2.1.7.2.2.3 Upon receipt of a CM-logon service indication, the CM-ground-user shall create the actual TSAP for each aircraft application information contained in the *Logon Request* based on the IDP and long TSAP for each application as defined in 2.1.4.

Note.— The actual TSAP = IDP + long TSAP. The IDP = AFI + IDI.

2.1.7.2.2.4 Upon the receipt of a *Logon Request* from a CM-logon service indication from an aircraft for which CM information has previously been received and still being maintained, the CM-ground-user shall update the aircraft information accordingly.

2.1.7.2.2.5 Upon receipt of a CM-logon service indication, the CM-ground-user shall invoke a CM-logon service response with a CMLogonResponse containing:

a) application names, addresses, and version numbers for the requested applications that can be air-initiated for all versions that the ground and aircraft systems can support, and

b) application names and version numbers for the requested ground-only initiated applications that the ground system can support.

2.1.7.2.2.6 When invoking the CM-logon service response, if any RDP for a given application address is different than the CM RDP, the CM-ground-user shall use the long TSAP for each application address provided.

Note 1.— The long TSAP = RDP + short TSAP. The short TSAP = ARS (optional) + LOC + SYS + NSEL +TSEL. The RDP = VER + ADM + RDF.

Note 2.— If there is more than one routing domain on the ground, the ARS field is used to differentiate them. If there is not more than one routing domain on the ground, the ARS field need not be used.

Note 3.— The value of the ARS field is a 24-bit unsigned binary number that uniquely identifies the addressed system in a single routing domain and is assigned by the State or Organization identified in the ADM field.

2.1.7.2.2.7 When the CM-ground-user requires a CM dialogue to be maintained, the CM-ground-user shall set the CM-logon service response *Maintain Dialogue* parameter, if and only if the dialogue maintain service is supported.

2.1.7.2.3 CM-update Service Requirements

Note.— Only the CM-ground-user is permitted to initiate the CM-update-service.

2.1.7.2.3.1 When invoking the CM-update service request, the CM-ground-user shall provide a CMUpdate containing application names, addresses, and version numbers for each of the data link applications being updated.

2.1.7.2.3.2 When invoking the CM-update service request, the CM-ground-user shall use the Long TSAP for each application address provided.

2.1.7.2.4 CM-contact Service Requirements

Note.— Only the CM-ground-user is permitted to initiate the CM-contact-service.

2.1.7.2.4.1 When invoking the CM-contact service request, the CM-ground-user shall provide a CMContactRequest containing the facility designation of the ground facility that the ground requests the aircraft to contact.

2.1.7.2.5 CM-end Service Requirements

Note 1.— Only the CM-ground-user is permitted to initiate the CM-end-service.

Note 2.— If the CM-ground-user establishes a CM dialogue with the CM-logon Maintain Dialogue *parameter set, the CM-ground user is responsible for closing the CM dialogue with the CM-end service.*

2.1.7.2.6 CM-forward Service Requirements

Note.— Only the CM-ground-user is permitted to initiate the CM-forward-service.

2.1.7.2.6.1 When requesting the CM-forward service, the CM-ground-user shall provide all of the information from either a CM-logon request message or a CM-forward request message, whichever is the more recent.

2.1.7.2.6.2 Upon receipt of a CM-forward service indication, the CM-ground-user shall make the aircraft application information contained in the *Forward Request* available to the other applications (i.e., ADS, CPDLC, and FIS), as well as to the dialogue service provider.

2.1.7.2.6.3 Upon receipt of a CM-forward service indication, the CM-ground-user shall create the actual TSAP for each aircraft application information contained in the *Forward Request* based on the IDP and long TSAP for each application as defined in 2.1.4.

Note.— The actual TSAP =IDP + long TSAP. The IDP = AFI + IDI.

2.1.7.2.6.4 Upon the receipt of a *Forward Request* from a CM-forward service indication concerning an aircraft identifier for which CM information has previously been received and is still being maintained, the CM-ground-user shall update the aircraft information accordingly.

2.1.7.2.6.5 Upon the receipt of a *Forward Request* from a CM-forward service indication, the receiving CM-ground-user shall invoke a CM-update service request with the indicated aircraft.

2.1.7.3 Parameter Value Unit, Range and Resolution

2.1.7.3.1 A CM user shall interpret parameter value unit, range and resolution as defined in 2.1.4.

2.1.8 SUBSETTING RULES

2.1.8.1 General

Note.— 2.1.8.1.1 specifies conformance requirements which all implementations of the CM protocol obey.

2.1.8.1.1 An implementation of either the CM ground based service or the CM air based service claiming conformance shall support the CM protocol features as shown in the table below.

Note 1.— The 'status' column indicates the level of support required for conformance to the CM-ASE protocol. The values are as follows:

a) ***'M'*** *mandatory support is required,*

b) ***'O'*** *optional support is permitted for conformance to the CM protocol,*

c) ***'N/A'*** *the item is not applicable, and*

d) ***'C.n'*** *the item is conditional where n is the number which identifies the condition which is applicable.*

Table 2.1.8-1. CM Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 2.1.8-2. CM Protocol Options

	Status	**Associated Predicate**
CM-air-ASE	C.1	CM/air
CM-ground-ASE	C.1	CM/ground
CM update supported	if (CM/ground) O, else M	UP-FU
CM contact supported	if (CM/ground) O, else M	CO-FU
CM maintain dialogue supported	if (CM/ground) O, else M	MA-FU

	Status	**Associated Predicate**
CM forward receiving user supported	if (CM/ground) O, else N/A	FO-FU
CM forward initiating user supported	if (CM/ground) O, else N/A	FO-IN

C.1: a conforming implementation will support one and only one of these two options.

Table 2.1.8-3. CM-ground-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I	CM/ground	CM-ground-ASE supporting logon, forward response and abort only. This represents the core functionality.
II	CM/ground + UP-FU	Core functionality and update only.
III	CM/ground + CO-FU	Core functionality and contact only.
IV	CM/ground + CO-FU + UP-FU	Core functionality, contact and update only.
V	CM/ground + UP-FU + MA-FU	Core functionality, update and dialogue only.
VI	CM/ground + CO-FU + MA-FU	Core functionality contact and dialogue only.
VII	CM/ground + CO-FU + UP-FU + MA-FU	Core functionality, contact, update and dialogue only.
VIII	CM/ground + FO-FU	Core functionality and forward user only.
IX	CM/ground + FO-FU + UP-FU	Core functionality, forward user and update only.
X	CM/ground + FO-FU + CO-FU	Core functionality, forward user and contact only.
XI	CM/ground + CO-FU + FO-FU + UP-FU	Core functionality, contact, forward user and update only.
XII	CM/ground + UP-FU + FO-FU + MA-FU	Core functionality, update, forward user and dialogue only.
XIII	CM/ground + CO-FU + FO-FU + MA-FU	Core functionality contact, forward user and dialogue only.

	List of Predicates	**Functionality Description**
XIV	CM/ground + CO-FU + UP-FU + FO-FU + MA-FU	Core functionality, contact, update, forward user and dialogue only.
XV	CM/ground + FO-IN	Core functionality and forward initiator only.
XVI	CM/ground + FO- IN + UP-FU	Core functionality, forward initiator and update only.
XVII	CM/ground + FO- IN + CO-FU	Core functionality, forward initiator and contact only.
XVIII	CM/ground + CO-FU + FO- IN + UP-FU	Core functionality, contact, forward initiator and update only.
XIX	CM/ground + UP-FU + FO- IN + MA-FU	Core functionality, update, forward initiator and dialogue only.
XX	CM/ground + CO-FU + FO- IN + MA-FU	Core functionality contact, forward initiator and dialogue only.
XXI	CM/ground + CO-FU + UP-FU + FO- IN + MA-FU	Core functionality, contact, update, forward initiator and dialogue only.
XXII	CM/ground + FO-FU + FO-IN	Core functionality and forward user and initiator only.
XXIII	CM/ground + UP-FU+ FO-FU + FO-IN	Core functionality, update and forward user and initiator only.
XXIV	CM/ground + CO-FU+ FO-FU + FO- IN	Core functionality, contact, and forward user and initiator only.
XXV	CM/ground + CO-FU + UP-FU+ FO-FU + FO- IN	Core functionality, contact, update and forward user and initiator only.

	List of Predicates	**Functionality Description**
XXVI	CM/ground + UP-FU + MA-FU+ FO-FU + FO- IN	Core functionality, update, dialogue and forward user and initiator only.
XXVII	CM/ground + CO-FU + MA-FU+ FO-FU + FO-IN	Core functionality, contact, dialogue and forward user and initiator only.
XXVIII	CM/ground + CO-FU + UP-FU + MA-FU+ FO-FU+ FO-IN	Core functionality, contact, update, dialogue and forward user and initiator (full functionality).

Note 2.— Capability to maintain a dialogue inherently includes the capability for a CM-ground user to choose not to maintain a dialogue.

Table 2.1.8-4. CM-air-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I	CM/air + CO-FU + UP-FU + MA-FU	CM-air-ASE supporting logon, contact, update, dialogue and abort (full functionality).

Table 2.1.8-5. Supported CM Service Primitives

	Sending (req,[cnf])	**Receiving (ind, [rsp])**
CM-logon	if (CM/air) M, else N/A	if (CM/ground) M, else N/A
CM-update	if (CM/ground and UP-FU) M, else N/A	if (CM/air) M, else N/A
CM-contact	if (CM/ground and CO-FU) M, else N/A	if (CM/air) M, else N/A
CM-end	if (CM/ground and MA-FU) M, else N/A	if (CM/air) M, else N/A
CM-forward	if (FO-IN) M, else N/A	if (FO-FU) M, else N/A
CM-user-abort	M	M
CM-provider-abort	N/A	M

Table 2.1.8-6. Supported CM APDUs

	Sender	**Receiver**
[CMAircraftMessage] cmLogonRequest	if (CM/air) M, else N/A	if (CM/ground) M, else N/A
[CMAircraftMessage] cmContactResponse	if (CM/air) M, else N/A	if (CM/ground and CO-FU) M, else N/A
[CMAircraftMessage] cmAbortReason	if (CM/air) M, else N/A	if (CM/ground) M, else N/A
[CMGroundMessage] cmLogonResponse	if (CM/ground) M, else N/A	if (CM/air) M, else N/A
[CMGroundMessage] cmUpdate	if (CM/ground and UP-FU) M, else N/A	if (CM/air) M, else N/A
[CMGroundMessage] cmContactRequest	if (CM/ground and CO-FU) M, else N/A	if (CM/air) M, else N/A
[CMGroundMessage] cmForwardRequest	if (FO-IN) M, else N/A	if (CM/ground) M, else N/A
[CMGroundMessage] cmForwardResponse	if (CM/ground) M, else N/A	if (FO-IN) M, else N/A
[CMGroundMessage]cmAbortReason	if (CM/ground) M, else N/A	M

2.2 AUTOMATIC DEPENDENT SURVEILLANCE APPLICATIONS

Note.— Structure of 2.2: 2.2.1 defines the air-ground communication aspects of ADS. 2.2.2 defines the ground-ground (i.e. ADS report forwarding) aspects of ADS.

2.2.1 AUTOMATIC DEPENDENT SURVEILLANCE APPLICATION

2.2.1.1 Introduction

2.2.1.1.1 The ADS air ground application will allow users to obtain positional and other information from suitably equipped aircraft in a timely manner in accordance with their requirements. The ADS application is designed to give automatic reports about aircraft to a user. The ADS reports give positional as well as other information likely to be of use to the air traffic management function, including air traffic control. The aircraft provides the information to the user under the following circumstances:

a) under a contract (known as a demand contract) agreed with the ground system, the aircraft provides the information immediately and once only;

b) under a contract (known as a periodic contract) agreed with the ground system, the aircraft provides information on a regular basis;

c) under a contract (known as an event contract) agreed with the ground system, the aircraft provides information when certain events are detected by the avionics;

d) under emergency conditions the aircraft provides information on a regular basis with no prior agreement with the ground system (known as an emergency contract).

Note 1.— Structure of 2.2: This chapter defines the air-ground communication aspects of ADS only.

a) 2.2.1.1: INTRODUCTION contains 2.2.1's purpose, structure, and a summary of the functions of ADS.

b) 2.2.1.2: GENERAL REQUIREMENTS contains backwards compatibility and error processing requirements.

c) 2.2.1.3: THE ABSTRACT SERVICE contains the description of the abstract service provided by the application service elements (ASE) defined for ADS.

d) 2.2.1.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by ADS-ASEs using Abstract Syntax Notation Number One (ASN.1).

e) 2.2.1.5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the ADS protocol, as well as time constraints and the exception handling procedures associated with these exchanges. 2.2.1 describes also the ADS protocol in terms of state tables.

f) 2.2.1.6: COMMUNICATION REQUIREMENTS contains the requirements that the ADS ASE application imposes on the underlying communication system.

g) 2.2.1.7: ADS USER REQUIREMENTS outlines the requirements that a user of an ADS ASE must meet.

h) 2.2.1.8: SUBSETTING RULES provides rules for subsetting the ADS SARPs.

Note 2.— General Functionality

a) The avionics are capable of supporting contracts with at least four ATC ground systems simultaneously; they are also capable of supporting one demand, one event and one periodic contract with each ground system simultaneously.

b) In addition if the pilot or avionics elects, the avionics will suspend any existing periodic contract, and establishes an emergency contract with each ground system with which it has an ADS contract.

c) It will be necessary for an implementation to provide information which is both accurate and timely in the ADS reports, however, quantification of the age and accuracy of the information is beyond the scope of 2.2.1.

Note 3.— Establishment and Operation of a Demand Contract

a) Functional Description

1) This function allows the ground system to establish a demand contract with an aircraft, and then for the conditions of that contract to be realised. Realisation of the contract involves the sending of a single report from an aircraft to the ground system.

2) Any number of demand contracts may be sequentially established with an aircraft. Basic information is sent with the report. Optionally, at the request of the ground system, other information may also be sent.

3) The ground system sends a demand contract request to the avionics. This contains an indication of which optional information blocks are required. The avionics then determines whether or not there are errors in the request, and if there are no errors, whether or not it is able to comply with the request. If the avionics can comply with the demand contract request it sends the report as soon as possible. If there are errors in the contract request, or if the avionics cannot comply with the request, it sends a negative acknowledgement to the ground system indicating the reason for its inability to accept the contract. If the avionics can partially comply with the request, it sends a non-compliance notification indicating those parts

of the contract with which it cannot comply, and then it sends an ADS-report.

b) Message Descriptions

1) The demand contract stipulates which of the optional information fields are to be included in the ADS report.

2) Each ADS-report always contains the following basic information:

i) the 3-D position of the aircraft;

ii) the time;

iii) an indication of the accuracy of the positional information (figure of merit).

3) Optionally, an ADS-report contains an indication of:

i) the aircraft address;

ii) the projected profile, indicating the position and predicted time of the next way point, and the position of the following way point;

iii) the ground vector, indicating the track, ground speed and vertical rate;

iv) the air vector, indicating the heading, air speed and vertical rate;

v) weather information, indicating wind speed, wind direction, temperature and turbulence;

vi) the short term intent, indicating the predicted location of the aircraft at some time in the future (as indicated in the demand contract) and, for any intermediate points where level, track or speed change is predicted to occur, the projected distance, track, level and time are given;

vii) extended project profile, indicating the predicted position, level and time for the next several way points (as indicated in the demand contract).

4) An ADS report can contain a positive acknowledgement indicating acceptance of the contract.

5) A negative acknowledgement contains an indication of the reason why the contract has not been accepted.

6) A non-compliance notification contains an indication of which optional information fields cannot be sent.

Note 4.— Establishment and Operation of an Event Contract

a) Functional Description

1) This function allows the ground system to establish an event contract with the aircraft, and then for the conditions of that contract to be realised. Realisation of the contract involves the sending of reports from the aircraft to the ground system when certain agreed events occur.

2) Only one event contract may exist between the ground system and avionics at any one time, but this may contain multiple event types. A set of basic information is sent with every report, and depending on the event that triggered the sending of the report, other information blocks may also be included. The contract that is agreed states the event types that are to trigger reports and also any values needed to clarify those event types.

3) It is possible to request one or more of the following event types:

i) Vertical rate change. This can be triggered in two ways. If the vertical rate threshold is positive, then the event is triggered when the aircraft's rate of climb is greater than the vertical rate threshold. If the vertical rate threshold is negative, then the event is triggered when the aircraft's rate of descent is less than the vertical rate threshold.

ii) Way-point change. This is triggered by a change to the next way-point. This change is normally due to routine way point sequencing, but could be triggered by a way point which is not part of the ATC clearance but is entered by the pilot for operational reasons.

iii) Lateral deviation change. This is triggered when the absolute value of the lateral distance between the aircraft's actual position and the aircraft's expected position on the active flight plan becomes greater than the lateral deviation threshold.

iv) Level range deviation. This is triggered when the aircraft's level becomes greater than the level ceiling or less than the level floor.

v) Airspeed change. This is triggered when the aircraft's airspeed differs negatively or positively from its value at the time of the

previous ADS report containing an air vector, by an amount which is equal to the airspeed change threshold which is specified in the event contract request. If there has been no previous such report, one is sent immediately.

vi) Ground speed change. This is triggered when the aircraft's ground speed differs negatively or positively from its value at the time of the previous ADS report containing a ground vector, by an amount which is equal to the ground speed threshold which is specified in the event contract request. If there has been no previous such report, one is sent immediately.

vii) Heading change. This is triggered when the aircraft's heading differs negatively or positively from its value at the time of the previous ADS report containing an air vector, by an amount which is equal to the heading change threshold which is specified in the event contract request. If there has been no previous such report, one is sent immediately.

viii) Extended projected profile change. This is triggered by a change to any of the set of future way points that define the active route of flight. The number of way points covered in the contract is either defined by a time interval (i.e. any way point planned to be achieved in the next N minutes), or by number of way points (i.e. any way point in the next N).

ix) FOM (Figure of Merit) change. This is triggered by a change in the navigational accuracy, navigational system redundancy or airborne collision avoidance system (ACAS) availability.

x) Track angle change. This is triggered when the aircraft's track angle differs negatively or positively from its value at the time of the previous ADS report containing a ground vector, by an amount which is equal to the track angle change threshold which is specified in the event contract request. If there has been no previous such report, one is sent immediately.

xi) Level change. This is triggered when the aircraft's level differs negatively or positively from its value at the time of the previous ADS report, by an amount which is equal to the level change threshold which is specified in the event contract request. If there has been no previous such report, one is sent immediately.

4) Acceptance of an event contract request implicitly cancels an existing event contract, if one exists.

5) The ground system sends an event contract request to the avionics. This contains the types of event to be reported on and the necessary parameters for that event (e.g. if the event is a level range deviation, then the upper and lower thresholds must be sent). The avionics then determines whether or not there are errors in the request, and if not, whether or not it is able to comply with the request. If the avionics can comply with the event contract request it sends a positive acknowledgement and any required baseline report. If the contracted event occurs, an ADS report is sent.

6) If there are errors in the event contract request, or if the avionics cannot comply with the request, it sends a negative acknowledgement to the ground system indicating the reason for its inability to accept the contract within 0.5 seconds.

7) If the avionics can partially comply with the request, it sends a non-compliance notification indicating those parts of the contract with which it cannot comply. If a contracted event occurs with which it can comply, an ADS-report is sent.

8) For lateral deviation, level range and vertical rate change, if the event occurs, a report is sent every 60 seconds while the limit(s) specified in the contract are exceeded. For all other events, a single report is sent every time the event occurs.

b) Message Descriptions

1) The event contract request contains an indication of the events to be reported on, together with clarifying information as follows:

i) lateral deviation change - containing the lateral deviation threshold;

ii) vertical rate change - containing the vertical rate threshold;

iii) leaving a given level range - containing the upper and lower level thresholds;

iv) way-point change - containing no further clarifying information;

v) air speed change - containing the airspeed change threshold;

vi) ground speed change - containing ground speed change threshold;

vii) heading change - containing heading change threshold;

viii) extended projected profile change - containing either a projected time or a number of way points;

ix) figure of merit change - containing no further clarifying information;

x) track angle change - containing the track angle change threshold;

xi) level change - containing level change range.

2) The ADS report has the same structure as in the operation of a demand contract, containing position, time and FOM. However the choice of additional optional information blocks is made as follows:

i) if the triggering event is a vertical rate change, a lateral deviation change, a level deviation change, a ground speed change, a track angle change or a level change, then the ADS report will contain the ground vector;

ii) if the triggering event is a way point change, then the ADS report will contain the projected profile;

iii) if the triggering event is an air speed change or heading change, then the ADS report will contain the air vector;

iv) if the triggering event is an extended projected profile change, then the ADS report will contain the extended projected profile;

v) if the triggering event is a FOM change, then the ADS report will contain no additional information other than the basic information contained in every ADS report).

3) An ADS report can contain a positive acknowledgement indicating acceptance of the contract.

4) A positive acknowledgement indicates acceptance of the contract and contains no further information.

5) A negative acknowledgement contains an indication of the reason why the message has not been accepted.

6) A non-compliance notification contains an indication of the events which the avionics cannot detect.

Note 5.— Establishment and Operation of a Periodic Contract

a) Functional Description

1) This function allows the ground system to establish a periodic contract with the aircraft, and then for the conditions of that contract to be realised. Realisation of the contract involves the sending of reports from the aircraft to the ground system at regular intervals (the reporting rate).

2) Only one periodic contract may exist between a ground system and the avionics at any one time. A set of basic information is sent with every report. Optionally, at the request of the ground system, other information blocks may also be sent; they may only be sent at a time interval which is a multiple of the reporting rate. The contract that is agreed includes the reporting rate, the optional blocks of information to be sent and the rate at which they are to be sent.

3) The ground system sends a periodic contract request to the avionics. This contains the basic reporting rate and an indication of which optional information blocks are required and how often they are to be sent relative to the basic rate (i.e. every time, every second report, every third report etc.). The avionics then determines whether or not there are errors in the request, and if not, whether or not it is able to comply with the request. If the avionics can comply with the periodic contract request it sends its first report , and then sends other reports at the intervals requested. If it cannot send the first report within 0.5 seconds, it sends a positive acknowledgement first to indicate its acceptance of the contract.

4) Acceptance of a periodic contract request implicitly cancels any existing periodic contract.

5) If there are errors in the periodic contract request, or if the avionics cannot accept the contract, it sends a negative acknowledgement to the ground system indicating the reason for its inability to accept the contract within 0.5 seconds.

6) If the avionics can partially comply with the request, it sends a non-compliance notification indicating those parts of the contract with which it cannot comply. It then sends ADS-reports at a rate with which it can comply, and containing information requested with which it can comply. Non-compliance can be caused by either inability to meet the requested reporting rate, and/or inability to supply the requested information

b) Message Descriptions

1) The periodic contract request may optionally contain any of the following information:

i) reporting interval;

ii) aircraft address modulus;

iii) projected profile modulus;

iv) ground vector modulus;

v) air vector modulus;

vi) weather modulus;

vii) short term intent modulus and projection time;

viii) extended projected profile modulus.

ix) Moduli indicate the multiple of the reporting rate that the information block is sent at (e.g. weather modulus of 5 means that the weather information block is sent with every 5th report).

2) The ADS report has the same structure as in the operation of a demand.

3) An ADS report can contain a positive acknowledgement indicating acceptance of the contract.

4) A positive acknowledgement indicates acceptance of the contract and contains no further information.

5) A negative acknowledgement contains an indication of the reason why the contract has not been accepted.

6) A non-compliance notification contains an indication of which optional information fields cannot be sent, and/or indicates that the requested periodic report cannot be met

Note 6.— Cancellation of Contracts

a) Functional Description

1) This function allows the ground system explicitly to cancel a contract that is in operation. The ground system sends a cancel contract message to the

avionics. The avionics cancels the contract and acknowledges the cancellation.

2) Implicit cancellation occurs when a periodic contract is in place, and then the ground system establishes a new periodic contract - the first one is implicitly cancelled on the establishment of the second; similarly with event contracts. Demand contracts are implicitly cancelled when the report is sent. There are no additional information flows associated with implicit cancellation.

3) The ground system may also cancel all contracts in a single cancel all contracts message. The avionics cancels all contracts and acknowledges the cancellation.

b) Message Descriptions

1) The cancel contract message contains an indication of the contract to be cancelled.

2) The cancel all contracts message contains no additional information.

3) A positive acknowledgement contains no additional information

Note 7.— Establishment and Operation of Emergency Contracts

a) Functional Description

1) This function allows the avionics to initiate an emergency contracts (either on instruction from the pilot or on its own initiative), between the avionics and all ground systems with which it has existing contracts. Realisation of the contract involves the sending of ADS emergency reports from the avionics to the ground system at regular intervals

2) Any existing periodic contract is suspended pending the cancellation of the emergency contract. Initially, the emergency reporting rate is the lesser of 60 seconds or half any existing periodic contract rate (if one exists).

3) The avionics sends ADS-emergency-reports to the ground system at the emergency reporting rate.

4) The avionics sends ADS-emergency-reports to all ground systems with which it has event or periodic contracts.

b) Message Descriptions

1) Each ADS-emergency-report always contains the following basic information:

i) the 3-D position of the aircraft;

ii) the time;

iii) an indication of the accuracy of the positional information (figure of merit).

2) With every fifth ADS-emergency-report, the following information is also included:

i) the aircraft address;

ii) the ground vector, indicating the track, ground speed and vertical rate;

Note 8.— Modifying an Emergency Contract

a) Functional Description

1) This function allows the reporting rate of an emergency contract to be modified.

2) The ground system sends an emergency contract modification message to the avionics. The avionics modifies the reporting rate of the emergency contract, and then sends the emergency reports at the new interval. This only effects the emergency contract between the ground system making the request and the aircraft.

3) If the avionics is unable to change the reporting rate, the avionics will send a negative acknowledgement within 0.5 seconds.

b) Message Descriptions

1) The emergency contract modification message contains only a new reporting rate.

2) A negative acknowledgement will contain an indication that the reporting rate cannot be changed.

Note 9.— Cancellation of Emergency Contracts

a) Functional Description

1) This function allows the aircraft to cancel an emergency contract.

2) The avionics sends a cancel emergency contract message to the ground system and cancels the emergency contract. If there is an periodic contract in place when the emergency is cancelled, then it is reinstated. Emergency contract cancellation cancels all emergency contracts.

b) Message Descriptions

1) The cancel emergency contract message contains no information.

2.2.1.2 General Requirements

2.2.1.2.1 ADS ASE Version Number

2.2.1.2.1.1 The ADS-air-ASE and the ADS-ground-ASE version numbers shall both be set to one.

2.2.1.2.2 Error Processing Requirements

2.2.1.2.2.1 In the event of information input by the ADS-user being incompatible with that able to be processed by the system, the user shall be notified.

2.2.1.2.2.2 In the event of an ADS-user invoking an ADS service primitive, when the ADS-ASE is not in a state specified in 2.2.1.5, the user shall be notified.

2.2.1.3 The Abstract Service

2.2.1.3.1 Service Description

2.2.1.3.1.1 An implementation of either the ADS ground based service or the ADS air based service shall exhibit behaviour consistent with having implemented an ADS-ground-ASE, or ADS-air-ASE respectively.

Note 1.— 2.2.1.3 defines the abstract service interface for the ADS service. The ADS-ASE abstract service is described in 2.2.1.3 from the viewpoint of the ADS-ASE-air-user, the ADS-ASE-ground-user and the ADS service-provider.

Note 2.— 2.2.1.3 defines the static behaviour (i.e. the format) of the ADS abstract service. Its dynamic behaviour (i.e. how it is used) is described in 2.2.1.7.

Note 3.— Figure 2.2.1.3-1 shows the functional model of the ADS Application. The functional modules identified in this model are the following :

a) the ADS-user,

b) the ADS Application Entity (ADS-AE) service interface,

c) the ADS-AE,

d) the ADS Control Function (ADS-CF),

e) the ADS Application Service Element (ADS-ASE) service interface,

f) the ADS-ASE, and

g) the Dialogue Service (DS) interface.

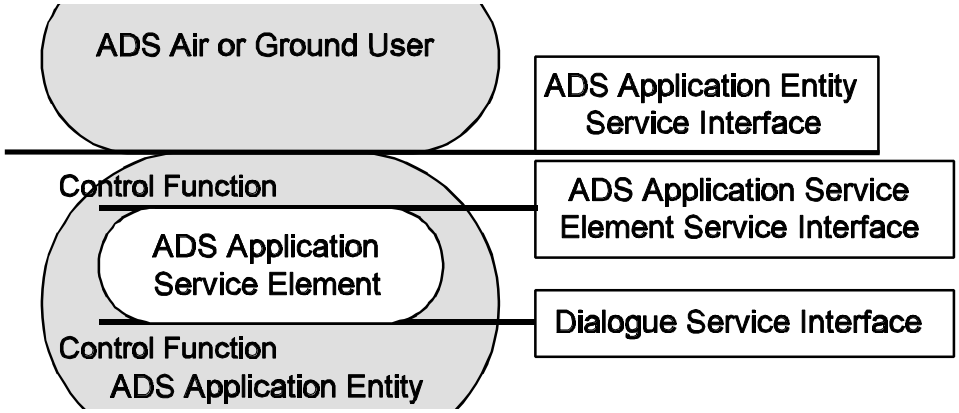

Figure 2.2.1.3-1 : Functional Model of the ADS Application

Note 4.— The ADS-user represents the operational part of the ADS system. This user does not perform the communication functions but relies on a communication service provided to it via the ADS-AE through the ADS-AE service interface. The individual actions at this interface are called ADS-AE service primitives. Similarly, individual actions at other interfaces in the communication system are called service primitives at these interfaces.

Note 5.— The ADS-AE consists of several elements, including the ADS-ASE and the ADS-CF. The DS interface is made available by the ADS-CF to the ADS-ASE for communication with the peer ADS-ASE.

Note 6.— The ADS-ASE is the element in the communication system which executes the ADS specific protocol. In other words, it takes care of the ADS specific service primitive sequencing actions, message creation, timer management, error and exception handling.

Note 7.— The ADS-ASE interfaces only with the ADS-CF. This ADS-CF is responsible for mapping service primitives received from one element (such as the ADS-ASE and the ADS-user) to other elements which interface with it. The part of the ADS-CF which is relevant from the point of view of these SARPs, i.e. the part between the ADS-user and the ADS-ASE, will map ADS-AE service primitives to ADS-ASE service primitives transparently.

Note 8.— The DS interface is the interface between the ADS-ASE and part of ADS-CF underneath, the ADS-ASE and provides the dialogue service.

2.2.1.3.2 The ADS-ASE Abstract Service

Note.— There is no requirement to implement the service in an ADS product; however, it is necessary to implement the ground based and air based system in such a way that it will be impossible to detect (from the peer system) whether or not an interface has been built.

2.2.1.3.2.1 The ADS-ASE abstract service shall consist of a set of the following services as allowed by the subsetting rules defined in 2.2.1.8:

a) *ADS-demand-contract* service as defined in 2.2.1.3.4;

b) *ADS-event-contract* service as defined in 2.2.1.3.5;

c) *ADS-periodic-contract* service as defined in 2.2.1.3.6;

d) *ADS-report* service as defined in 2.2.1.3.7;

e) *ADS-cancel* service as defined in 2.2.1.3.8;

f) *ADS-cancel-all-contracts* service as defined in 2.2.1.3.9;

g) *ADS-emergency-report* service as defined in 2.2.1.3.10;

h) *ADS-modify-emergency-contract* service as defined in 2.2.1.3.11;

i) *ADS-cancel-emergency* service as defined in 2.2.1.3.12;

j) *ADS-user-abort* service as defined in 2.2.1.3.13;

k) *ADS-provider-abort* service as defined in 2.2.1.3.14.

Note 1.— ADS-demand-contract, ADS-event-contract, ADS-periodic-contract, ADS-cancel, ADS-cancel-all-contracts and ADS-modify-emergency-contract are only initiated by the ADS-ground-user.

Note 2.— ADS-report, ADS-emergency-report and ADS-cancel-emergency are only initiated by the ADS-air-user.

Note 3.— ADS-user-abort is initiated by an ADS-air-user or an ADS-ground-user.

Note 4.— ADS-provider-abort is only initiated by the ADS-service-provider.

Note 5.— An abstract syntax is a syntactical description of a parameter which does not imply a specific implementation. Only when the ADS-ASE maps a parameter onto an APDU field, or vice-versa, is the abstract syntax of the parameter described by using the ASN.1 of 2.2.1.4 for this field

2.2.1.3.3 Conventions

Note 1.— For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables 2.2.1.3:

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.*

Note 2.— The following abbreviations are used in this document:

a)	*Req*	*request; data is input by an ADS user initiating the service to its respective ASE;*
b)	*Ind*	*indication; data is indicated by the receiving ASE to its respective ADS user;*
c)	*Rsp*	*response; data is input by receiving ADS user to its respective ASE;*
d)	*Cnf*	*confirmation; data is confirmed by the initiating ASE to its respective ADS user.*

Note 3.— An unconfirmed service allows just one message to be transmitted, in one direction.

Note 4.— A confirmed service provides end-to-end confirmation that a message sent by one user was received by its peer user.

2.2.1.3.4 ADS-demand-contract Service

Note.— The ADS-demand-contract service allows the ADS-ground-user to request a demand contract with the aircraft. It is a confirmed service, initiated by the ADS-ground-user.

2.2.1.3.4.1 The ADS-demand-contract service shall contain primitives and parameters as presented in Table 2.2.1.3-1, when an acknowledgement is sent independent of an ADS-report.

Table 2.2.1.3-1: ADS-demand-contract service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Aircraft identifier	M			
Class of communication service	U			
Contract details	M	M(=)		
Reply			M	M(=)
ICAO facility designation	C	C(=)		

2.2.1.3.4.2 The ADS-demand-contract service primitives shall contain the parameters as presented in Table 2.2.1 3-2, when a positive acknowledgement is embedded in an ADS-report.

Table 2.2.1.3-2: ADS-demand-contract service parameters

Parameter Name	Req	Ind
Aircraft identifier	M	
Class of communication service	U	
Contract details	M	M(=)
ICAO facility designation	C	C(=)

2.2.1.3.4.3 Aircraft Identifier

Note.— This parameter contains the 24 bit ICAO address of the aircraft with which the contract is being made.

2.2.1.3.4.3.1 The *aircraft identifier* parameter value shall conform to an abstract value corresponding to a 24-bit ICAO aircraft-id.

2.2.1.3.4.4 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service, if specified by the ADS-ground-user.

2.2.1.3.4.4.1 Where specified by the ADS-ground-user, the *class of communication service* parameter shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G” or “H”.

Note 1.— If contracts are currently in place, the class of communication service parameter is not used by the ADS-service provider.

Note 2.— Where not specified by the ADS-ground-user, when there are no contracts already in force, this indicates that there will be no routing preference.

2.2.1.3.4.5 Contract Details

Note.— This parameter contains the details of the contract as requested by the ADS-ground-user.

2.2.1.3.4.5.1 The *contract details* parameter value shall conform to the ASN.1 abstract syntax *DemandContract*.

2.2.1.3.4.6 Reply

Note 1.— This parameter indicates the extent to which the contract request can be complied with. If it has the value NegativeAcknowledgement, it indicates that the contract has been rejected and gives reasons. If it has the value NoncomplianceNotification, it indicates that only some parts of the contract can be complied with, and indicates which ones have been rejected.

Note 2.— Unlike Event and Periodic contracts, the “Reply” parameter does not have the option of containing a positive acknowledgement.

2.2.1.3.4.6.1 The *reply* parameter value shall conform to one of the following abstract syntaxes:

a) The abstract value “Negative Acknowledgement”, and a value conforming to the ASN.1 abstract syntax *Reason*; or

b) The abstract value “Noncompliance Notification”, and a value conforming to the ASN.1 abstract syntax *NoncomplianceNotification* with the choice *demand-ncn*.

2.2.1.3.4.7 ICAO facility designation

Note.— this parameter contains the 4 to 8 character ICAO facility designation of the ICAO facility which is initiating the contract. If contracts are currently in place, this parameter is not used by the ADS-service-provider.

2.2.1.3.4.7.1 The *ICAO facility designation* parameter value shall conform to an abstract value corresponding to a 4 to 8 character ICAO facility designation.

2.2.1.3.4.7.2 The *ICAO facility designation* parameter value shall be provided when the ADS-ground-user has no other contracts in place with the aircraft.

2.2.1.3.5 ADS-event-contract Service

Note.— The ADS-event-contract service allows the ADS-ground-user to request an event contract with the aircraft It is a confirmed service, initiated by the ADS-ground-user.

2.2.1.3.5.1 The ADS-event-contract service shall contain primitives and parameters as presented in Table 2.2.1.3-3, when an acknowledgement is sent independent of an ADS-report .

Table 2.2.1.3-3: ADS-event-contract service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Aircraft identifier	M			
Class of communication service	U			
Contract details	M	M(=)		
Reply			M	M(=)
ICAO facility designation	C	C(=)		

2.2.1.3.5.2 The ADS-event-contract service shall contain primitives and parameters as presented in Table 2.2.1.3-4, when a positive acknowledgement is embedded in an ADS-report.

Table 2.2.1.3-4: ADS-event-contract service parameters

Parameter Name	Req	Ind
Aircraft identifier	M	
Class of communication service	U	
Contract details	M	M(=)
ICAO facility designation	C	C(=)

2.2.1.3.5.3 Aircraft Identifier

Note.— This parameter contains the 24 bit ICAO address of the aircraft with which the contract is being made.

2.2.1.3.5.3.1 The *aircraft identifier* parameter value shall conform to an abstract value corresponding to a 24-bit ICAO aircraft-id.

2.2.1.3.5.4 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service, if specified by the ADS-ground-user.

2.2.1.3.5.4.1 Where specified by the ADS-ground-user, the *class of communication service* parameter shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G” or “H”.

Note 1.— If contracts are currently in place, the class of communication service parameter is not used by the ADS-service provider.

Note 2.— Where not specified by the ADS-ground-user, when there are no contracts already in force, this indicates that there will be no routing preference.

2.2.1.3.5.5 Contract Details

Note.— This parameter contains the details of the contract as requested by the ADS-ground-user.

2.2.1.3.5.5.1 The *contract details* parameter value shall conform to the ASN.1 abstract syntax *EventContract*.

2.2.1.3.5.6 Reply

Note.— This parameter indicates the extent to which the contract request can be complied with. If it has the value NegativeAcknowledgement, it indicates that the contract has been rejected and gives reasons. If it has the value NoncomplianceNotification, it indicates that only some parts of the contract can be complied with, and indicating which ones have been rejected. If it has the value PositiveAcknowledgement, it indicates full compliance with the contract has been accepted.

2.2.1.3.5.6.1 The *reply* parameter value shall conform to one of the following abstract syntaxes:

a) The abstract value “Negative Acknowledgement”, and a value conforming to the ASN.1 abstract syntax *Reason*;

b) The abstract value “Noncompliance Notification”, and a value conforming to the ASN.1 abstract syntax *NoncomplianceNotification* with the choice event-ncn;

c) The abstract value “Positive Acknowledgement”, and a value conforming to the ASN.1 abstract syntax *NULL*.

2.2.1.3.5.7 ICAO facility designation

Note.— This parameter contains the 4 to 8 character ICAO facility designation of the ICAO facility which is initiating the contract. If contracts are currently in place, this parameter is not used by the ADS-service-provider.

2.2.1.3.5.7.1 The *ICAO facility designation* parameter value shall conform to an abstract value corresponding to a 4 to 8 character ICAO facility designation.

2.2.1.3.5.7.2 The *ICAO facility designation* parameter value shall be provided when the ADS-ground-user has no other contracts in place with the aircraft.

2.2.1.3.6 ADS-periodic-contract Service

Note.— The ADS-periodic-contract service allows the ADS-ground-user to request a periodic contract with the aircraft. It is a confirmed service, initiated by the ADS-ground-user.

2.2.1.3.6.1 The ADS-periodic-contract service shall contain primitives and parameters as presented in Table 2.2.1.3-5, when an acknowledgement is sent independent of an ADS-report.

Table 2.2.1.3-5: ADS-periodic-contract service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Aircraft identifier	M			
Class of communication service	U			
Contract details	M	M(=)		
Reply			M	M(=)
ICAO facility designation	C	C(=)		

2.2.1.3.6.2 The ADS-periodic-contract service shall contain primitives and parameters as presented in Table 2.2.1.3-6, when a positive acknowledgement is embedded in an ADS-report.

Table 2.2.1.3-6: ADS-periodic-contract service parameters

Parameter Name	Req	Ind
Aircraft identifier	M	
Class of communication service	U	
Contract details	M	M(=)
ICAO facility designation	C	C(=)

2.2.1.3.6.3 Aircraft Identifier

Note.— This parameter contains the 24 bit ICAO address of the aircraft with which the contract is being made.

2.2.1.3.6.3.1 The *aircraft identifier* parameter value shall conform to an abstract value corresponding to a 24-bit ICAO aircraft-id.

2.2.1.3.6.4 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service, if specified by the ADS-ground-user.

2.2.1.3.6.4.1 Where specified by the ADS-ground-user, the *class of communication service* parameter shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G” or “H”.

Note 1.— If contracts are currently in place, the class of communication service parameter is not used by the ADS-service provider.

Note 2.— Where not specified by the ADS-ground-user, when there are no contracts already in force, this indicates that there will be no routing preference.

2.2.1.3.6.5 Contract Details

Note.— This parameter contains the details of the contract as requested by the ADS-ground-user.

2.2.1.3.6.5.1 The *contract details* parameter value shall conform to the ASN.1 abstract syntax *PeriodicContract*.

2.2.1.3.6.6 Reply

Note.— This parameter indicates the extent to which the contract request can be complied with. If it has the value Negative Acknowledgement, it indicates that the contract has been rejected and gives reasons. If it has the value NoncomplianceNotification, it indicates that only some parts of the contract can be complied with, and indicating what part of the contract cannot be conformed to. If it has the value PositiveAcknowledgement, it indicates that the contract has been accepted but cannot be satisfied immediately, either because there is an emergency contract in place, or because the information is not available within the 0.5 second turnaround time.

2.2.1.3.6.6.1 The *reply* parameter value shall conform to one of the following abstract syntaxes:

a) The abstract value “Negative Acknowledgement”, and a value conforming to the ASN.1 abstract syntax *Reason*;

b) The abstract value “Noncompliance Notification”, and a value conforming to the ASN.1 abstract syntax *NoncomplianceNotification* with the choice *periodic-ncn*;

c) The abstract value “Positive Acknowledgement”, and a value conforming to the ASN.1 abstract syntax *NULL*.

2.2.1.3.6.7 ICAO facility designation

Note.— This parameter contains the 4 to 8 character ICAO facility designation of the ICAO facility which is initiating the contract. If contracts are currently in place, this parameter is not used by the ADS-service-provider.

2.2.1.3.6.7.1 The *ICAO facility designation* parameter value shall conform to an abstract value corresponding to a 4 to 8 character ICAO facility designation.

2.2.1.3.6.7.2 The *ICAO facility designation* parameter value shall be provided when the ADS-ground-user has no other contracts in place with the aircraft.

2.2.1.3.7 ADS-report Service

Note.— The ADS-report service allows the ADS-air-user to send an ADS report to the ADS-ground-user. This is an unconfirmed service, initiated by the ADS-air-user.

2.2.1.3.7.1 The ADS-report service shall contain primitives and parameters as contained in Table 2.2.1.3-7.

Table 2.2.1.3-7: ADS-report service parameters

Parameter Name	Req	Ind
Contract type	M	M(=)
Event Type	C	C(=)
Positive Acknowledgement	U	C(=)
Report details	M	M(=)

2.2.1.3.7.2 Contract Type

Note.— This parameter identifies the type of contract that this report is in response to.

2.2.1.3.7.2.1 The *contract type* parameter value shall contain one of the abstract values “demand contract”, “event contract”, or “periodic contract”.

2.2.1.3.7.3 Event Type

Note.— This parameter indicates the type of event that triggered the report.

2.2.1.3.7.3.1 The *event type* parameter shall be present if, and only if, the *contract type* parameter has the abstract value event-contract.

2.2.1.3.7.3.2 The *event type* parameter shall contain a value conforming to the abstract syntax *EventTypeReported*.

2.2.1.3.7.4 Positive Acknowledgement

Note.— This parameter is used to indicate that the report carries with it a positive acknowledgement of a new event contract, or a new periodic contract, or a new demand contract.

2.2.1.3.7.4.1 The *positive acknowledgement* parameter abstract syntax shall be NULL.

2.2.1.3.7.5 Report Details

Note.— This parameter contains the details of the ADS report.

2.2.1.3.7.5.1 The *report details* parameter value shall conform to the ASN.1 abstract syntax *ADSReport*.

2.2.1.3.8 ADS-cancel Service

Note.— The ADS-cancel service allows the ADS-ground-user to cancel an existing contract. It is a confirmed service, initiated by the ADS-ground-user.

2.2.1.3.8.1 The ADS-cancel service shall contain primitives and parameters as contained in Table 2.2.1.3-8.

Table 2.2.1.3-8: ADS-cancel service parameters

Parameter Name	Req	Ind	Cnf
Contract type	M	M(=)	M(=)

2.2.1.3.8.2 Contract Type

Note.— This parameter identifies the type of contract that is to be cancelled.

2.2.1.3.8.2.1 The *contract type* parameter value shall conform to the ASN.1 abstract syntax *CancelContract*.

2.2.1.3.9 ADS-cancel-all-contracts Service

Note.— The ADS-cancel-all-contracts service allows the ADS-ground-user to cancel all contracts with a particular aircraft. It is a confirmed service, initiated by the ADS-ground-user.

2.2.1.3.9.1 The ADS-cancel-all-contracts service shall contain primitives as contained in Table 2.2.1.3-9.

Table 2.2.1.3-9: ADS-cancel-all-contracts service parameters

Parameter Name	Req	Ind	Cnf
none			

2.2.1.3.10 ADS-emergency-report Service

Note.— The ADS-emergency-report service allows the ADS-air-user to send an emergency ADS report to the ADS-ground-user. This is an unconfirmed service, initiated by the ADS-air-user.

2.2.1.3.10.1 The ADS-emergency-report service shall contain primitives and parameters as contained in Table 2.2.1.3-10.

Table 2.2.1.3-10: ADS-emergency-report service parameters

Parameter Name	Req	Ind
Positive acknowledgement of modification	U	C(=)
Emergency report details	M	M(=)

2.2.1.3.10.2 Positive Acknowledgement of Modification

Note.— The presence of this parameter indicates that the ADS-air-user has received the ADS-modify-emergency-contract indication and has accepted it.

2.2.1.3.10.2.1 The *positive acknowledgement of modification* parameter abstract syntax shall be *NULL*.

2.2.1.3.10.3 Emergency report details

Note.— The parameter contains the details of the emergency report.

2.2.1.3.10.3.1 The *emergency report details* parameter value shall conform to the ASN.1 abstract syntax *ADSEmergencyReport*.

2.2.1.3.11 ADS-modify-emergency-contract Service

Note.— The ADS-modify-emergency-contract service allows the ADS-ground-user to request changes to an emergency contract reporting rate. It is a confirmed service, initiated by the ADS-ground-user.

2.2.1.3.11.1 The ADS-modify-emergency-contract service shall contain primitives and parameters as contained in Table 2.2.1.3-11, when the request to modify the emergency rate is refused.

Table 2.2.1.3-11: ADS-modify-emergency-contract parameters

Parameter Name	Req	Ind	Rsp	Cnf
Reporting interval	M	M(=)		

2.2.1.3.11.2 The ADS-modify-emergency-contract service shall contain primitives and parameters as presented in Table 2.2.1.3-12, when the request to modify the emergency rate is accepted, and a positive acknowledgement is embedded in an ADS-emergency-report.

Table 2.2.1.3-12: ADS-modify-emergency-contract parameters

Parameter Name	Req	Ind
Reporting interval	M	M(=)

2.2.1.3.11.3 Reporting Interval

Note.— This parameter indicates the new interval for sending the ADS emergency reports.

2.2.1.3.11.3.1 The *reporting interval* parameter value shall conform to the ASN.1 abstract syntax *ReportingInterval*.

2.2.1.3.12 ADS-cancel-emergency Service

Note.— The ADS-cancel-emergency service allows the ADS-air-user to inform the ADS-ground-user that the emergency contract has been cancelled. When the emergency is concluded, the ADS-air-user must invoke this service with every ground system with which it has an emergency contract. This is an unconfirmed service, initiated by the ADS-air-user.

2.2.1.3.12.1 The ADS-cancel-emergency service shall contain primitives as contained in Table 2.2.1.3-13.

Table 2.2.1.3-13: ADS-cancel-emergency service parameters

Parameter Name	Req	Ind
none		

2.2.1.3.13 ADS-user-abort Service

Note 1.— The ADS-user-abort service allows the ADS-air-user to abort all ADS contracts with a particular ground system, or ADS-ground-user to abort all ADS contracts with a particular aircraft, or an ADS-ground-user to abort an ADS forward contract with a particular ground system. It is an unconfirmed service, initiated by an ADS-ground-user or the ADS-air-user. Messages in transit may be lost during this operation. It can be invoked at any time that the ADS-user is aware that any ADS service is in operation.

Note 2.— If the service is invoked prior to complete establishment of the dialogue, the ADS-user-abort indication may not be provided. An ADS-provider-abort indication may result instead.

2.2.1.3.13.1 The ADS-user-abort service shall contain primitives as contained in Table 2.2.1.3-14.

Table 2.2.1.3-14: ADS-user-abort service parameters

Parameter Name	Req	Ind
none		

2.2.1.3.14 ADS-provider-abort Service

Note.— The ADS-provider-abort service allows the ADS-service-provider to inform the ADS-ground-user and the ADS-air-user that it can no longer provide the ADS service for a particular ADS-ground-user - ADS-air-user pairing or a particular ADS-ground-user - ADS-ground-user pairing. It is initiated by the ADS-service-provider. Messages in transit may be lost during this operation.

2.2.1.3.14.1 The ADS-provider-abort service shall contain primitives and parameters as contained in Table 2.2.1.3-15.

Table 2.2.1.3-15: ADS-provider-abort service parameters

Parameter Name	Ind
Reason	M

2.2.1.3.14.2 Reason

Note.— This parameter identifies the reason for the abort.

2.2.1.3.14.2.1 The *reason* parameter shall conform to the ASN.1 abstract syntax *AbortReason*.

2.2.1.4 Formal Definitions of Messages

2.2.1.4.1 Encoding/Decoding Rules

2.2.1.4.1.1 An ADS-air-ASE shall be capable of decoding [ADSAircraftPDUs] APDUs and decoding [ADSGroundPDUs] APDUs.

2.2.1.4.1.2 An ADS-ground-ASE shall be capable of decoding [ADSGroundPDUs] APDUs and decoding [ADSAircraftPDUs] APDUs.

2.2.1.4.2 ADS ASN.1 Abstract Syntax

2.2.1.4.2.1 The abstract syntax of the air-ground ADS protocol data units shall comply with the description contained in the ASN.1 module ADSMessageSetVersion1 (conforming to ISO/IEC 882), as defined in 2.2.1.4.

Note .— Where units indicate directional information, the value is given relative to true North. If magnetic information is required this will be a matter for local ground implementation.

```
ADSMessageSetVersion1 DEFINITIONS::=

BEGIN

EXPORTS
        AbortReason, ADSEmergencyReport, ADSReport, AircraftAddress, EventTypeReported;

-- ------------------------------------------------------------------------------------------------------------
-- Aircraft-generated and Ground-generated Message Choice
-- ------------------------------------------------------------------------------------------------------------

ADSAircraftPDUs ::= CHOICE
{
        aDS-cancel-emergency-PDU                    [0]    NULL,
        aDS-demand-report-PDU                       [1]    ADSDemandReport,
        aDS-emergency-report-PDU                    [2]    ADSEmergency,
        aDS-event-report-PDU                        [3]    ADSEventReport,
        aDS-negative-acknowledgement-PDU            [4]    NegativeAcknowledgement,
        aDS-noncompliance-notification-PDU          [5]    NoncomplianceNotification,
        aDS-periodic-report-PDU                     [6]    ADSPeriodicReport,
        aDS-positive-acknowledgement-PDU            [7]    PositiveAcknowledgement,
        aDS-provider-abort-PDU                      [8]    AbortReason,
        ...
}

ADSGroundPDUs ::= CHOICE
{
        aDS-cancel-all-contracts-PDU                [0]    NULL,
        aDS-cancel-contract-PDU                     [1]    CancelContract,
        aDS-cancel-emergency-acknowledgement-PDU    [2]    NULL,
        aDS-demand-contract-PDU                     [3]    DemandContract,
```

```
        aDS-event-contract-PDU                  [4]     EventContract,
        aDS-modify-emergency-contract-PDU       [5]     ModifyEmergency,
        aDS-periodic-contract-PDU               [6]     PeriodicContract,
        aDS-provider-abort-PDU                  [7]     AbortReason,
        ...
}

-- ------------------------------------------------------------------------------------------------------------
-- Ground-generated and Aircraft-generated message components - Protocol Data Units
-- ------------------------------------------------------------------------------------------------------------

AbortReason  ::= ENUMERATED
{
        communications-service-failure  (0),
        unrecoverable-system-error              (1),
        invalid-PDU                             (2),
        sequence-error                          (3),
        timer-expiry                            (4),
        cannot-establish-contact                (5),
        undefined-error                         (6),
        dialogue-end-not-accepted               (7),
        unexpected-PDU                          (8),
        decoding-error                          (9),
        invalid-qos-parameter                   (10),
        ...
}

ADSDemandReport ::= SEQUENCE
{
        report                          [0]     ADSReport,
        positive-acknowledgement        [1]     NULL            OPTIONAL,
        ...
}

ADSEmergency ::= SEQUENCE
{
        emergency-report                [0]     ADSEmergencyReport,
        positive-acknowledgement        [1]     NULL            OPTIONAL,
        ...
}

ADSEventReport ::= SEQUENCE
{
        event-type                      [0]     EventTypeReported,
        report                          [1]     ADSReport,
        positive-acknowledgement        [2]     NULL            OPTIONAL,
```

```
        ...
}

ADSPeriodicReport ::= SEQUENCE
{
        report                          [0]     ADSReport,
        positive-acknowledgement        [1]     NULL                    OPTIONAL,
        ...
}

CancelContract ::= ENUMERATED
{
        event-contract                  (0),
        periodic-contract               (1),
        ...
}

DemandContract ::= SEQUENCE
{
        aircraft-address                [0]     NULL                    OPTIONAL,
        projected-profile               [1]     NULL                    OPTIONAL,
        ground-vector                   [2]     NULL                    OPTIONAL,
        air-vector                      [3]     NULL                    OPTIONAL,
        weather                         [4]     NULL                    OPTIONAL,
        short-term-intent               [5]     ProjectionTime  OPTIONAL,
        extended-projected-profile      [6]     ExtendedProjectedProfileRequest
                                                                        OPTIONAL,
        ...
}

EventContract ::= SEQUENCE
{

        lateral-deviation-change        [0]     LateralChange           OPTIONAL,
        vertical-rate-change                    [1]     VerticalRateChange      OPTIONAL,
        level-range                             [2]     LevelRange              OPTIONAL,
        way-point-change                        [3]     NULL                    OPTIONAL,
        air-speed-change                        [4]     AirSpeedChange          OPTIONAL,
        ground-speed-change                     [5]     GroundSpeedChange       OPTIONAL,
        heading-change                  [6]     DegreesDirection        OPTIONAL,
        extended-projected-profile-change       [7]     ExtendedProjectedProfileRequest
                                                                        OPTIONAL,
        fom-change                              [8]     NULL                    OPTIONAL,
        track-angle-change                      [9]     DegreesDirection        OPTIONAL,
        level-change                            [10]    LevelChange             OPTIONAL,
        ...
```

```
}

ModifyEmergency ::= ReportingInterval

NegativeAcknowledgement ::= SEQUENCE
{
        request-type                    RequestType,
        reason                          Reason
}

NoncomplianceNotification ::= CHOICE
{
        demand-ncn                [0]   SET OF ReportType,
        event-ncn                 [1]   SET OF EventTypeContracted,
        periodic-ncn              [2]   SET OF ReportTypeAndPeriod,
        ...
}

PeriodicContract ::= SEQUENCE
{
        reporting-interval                [0]   ReportingInterval
                                                        DEFAULT {minutes-scale 5},
        aircraft-address-modulus          [1]   Modulus                 OPTIONAL,
        projected-profile-modulus         [2]   Modulus                 OPTIONAL,
        ground-vector-modulus             [3]   Modulus                 OPTIONAL,
        air-vector-modulus                [4]   Modulus                 OPTIONAL,
        weather-modulus                   [5]   Modulus                 OPTIONAL,
        short-term-intent-modulus         [6]   ShortTermIntentModulus  OPTIONAL,
        extended-projected-profile-modulus [7]  ExtendedProjectedProfileModulus
                                                                        OPTIONAL,
        ...
}

PositiveAcknowledgement ::= RequestType

-- ------------------------------------------------------------------------------------------------------------
-- Reports and their components
-- ------------------------------------------------------------------------------------------------------------

ADSEmergencyReport ::= SEQUENCE
{
        position                  [0]   Position,
        time-stamp                [1]   DateTimeGroup,
        fom                       [2]   FigureOfMerit,
        aircraftAddress           [3]   AircraftAddress         OPTIONAL,
        ground-vector             [4]   GroundVector            OPTIONAL
```

```
}

ADSReport ::= SEQUENCE
{
        position                    [0]   Position,
        time-stamp                  [1]   DateTimeGroup,
        fom                         [2]   FigureOfMerit,
        aircraft-address            [3]   AircraftAddress          OPTIONAL,
        projected-profile           [4]   ProjectedProfile         OPTIONAL,
        ground-vector               [5]   GroundVector             OPTIONAL,
        air-vector                  [6]   AirVector                OPTIONAL,
        weather                     [7]   Weather                  OPTIONAL,
        short-term-intent           [8]   ShortTermIntent          OPTIONAL,
        extended-projected-profile  [9]   ExtendedProjectedProfile OPTIONAL,
        ...
}

AircraftAddress ::= BIT STRING (SIZE (24))
        -- 24 bit ICAO airframe identifier

AirVector ::= SEQUENCE
{
        heading                     [0]   DegreesDirection         OPTIONAL,
        air-speed                   [1]   AirSpeed                 OPTIONAL,
        vertical-rate               [2]   VerticalRateChange       OPTIONAL
}

AirSpeed ::= CHOICE
{
        mach                        [0]   Mach,
        ias                         [1]   Ias,
        mach-and-ias                [2]   SEQUENCE
                                          {
                                                Mach,
                                                Ias
                                          }
}

-- When AirSpeed is returned in an ADS report, the choice of which of the above units of
-- air speed are used depends on how the aircraft is equipped and whether the aircraft is
-- flying on Mach or IAS at the time. The choice is made by the avionics.


ExtendedProjectedProfile ::= SEQUENCE SIZE (1..128) OF SEQUENCE
{
        way-point                   Position,
```

```
        time                                    Eta
}

FigureOfMerit ::= SEQUENCE
{
        position-accuracy                       PositionAccuracy,
        multiple-navigational-units-operating   BOOLEAN,
        acas-operational                        BOOLEAN
}

PositionAccuracy ::= ENUMERATED   -- nm = nautical miles
{
        complete-loss   (0),
        under30nm       (1),
        under15nm       (2),
        under8nm        (3),
        under4nm        (4),
        under1nm        (5),
        under-25nm      (6),  -- under 0.25 nm
        under-05nm      (7)   -- under 0.05 nm
}

GroundVector ::= SEQUENCE
{
        track                   [0]   DegreesDirection              OPTIONAL,
        ground-speed            [1]   INTEGER (-50..2200)           OPTIONAL,
                                      -- units =knots
                                      -- range =-50 to +2200 knots
        vertical-rate           [2]   VerticalRateChange            OPTIONAL
}

ProjectedProfile ::= SEQUENCE
{
        next-way-point                  Position,
        next-time                       Eta,
        following-way-point             Position
}

ShortTermIntent ::= SEQUENCE
{
        position                        Position,
        projected-time                  ProjectionTime,
        intermediate-intent             IntermediateIntent
}

IntermediateIntent ::= SEQUENCE SIZE (0..7) OF SEQUENCE
```

```
{
        distance                        INTEGER (1..8000),
                                        -- units = Nautical miles
                                        -- range = 1 to 8000 Nautical miles
        track                           DegreesDirection,
        level                           Level,
        projected-time                  ProjectionTime
}

-- IntermediateIntent indicates a set of way points between the current position and the
-- time indicated in the ShortTerm intent element.
-- distance is expressed as relative to position at the time of the ADS-report.
-- track is expressed as absolute track.
-- level is expressed as absolute level.
-- projected-time is expressed as relative to the timestamp on the ADS-report.

Weather ::= SEQUENCE
{
        wind-speed              [0]     INTEGER (0..300) OPTIONAL,
                                        -- units = knots
                                        -- range = 0 to 300 knots
        wind-direction          [1]     INTEGER (1..360) OPTIONAL,
                                        -- units = degrees true North
                                        -- range= 1 to 360 degrees
        temperature             [2]     INTEGER (-400..400) OPTIONAL,
                                        -- units = 0.25 degrees Celsius
                                        -- range = -100 to 100 degrees C
        turbulence              [3]     INTEGER (0..15) OPTIONAL
                                        -- this is a place marker for a turbulence
                                        -- index which is to be defined
}

-- ------------------------------------------------------------------------------------------------------------
-- Components of Contracts
-- ------------------------------------------------------------------------------------------------------------

AirSpeedChange ::= CHOICE
{
        mach-number-change      [0]     INTEGER (1..255),
                                        -- units = 0.005 Mach
                                        -- range = 0.005 to 1.275 Mach
        ias-change              [1]     INTEGER (1..700)
                                        -- units = knots
                                        -- range = 1 to 700 knots
}
```

```
LevelChange ::= INTEGER (1..500)
        -- units =10 feet
        -- range =10 to 5 000 feet

LevelRange ::= SEQUENCE
{
        ceiling                         Level,
        floor                           Level
}

DegreesDirection ::= INTEGER (1..3600)
        -- units = 0.1 degrees true North,
        -- range = 0.1 to 360 degrees

ExtendedProjectedProfileModulus ::= SEQUENCE
{
        modulus                         Modulus,
        extended-projected-profile-request  ExtendedProjectedProfileRequest
}

ExtendedProjectedProfileRequest ::= CHOICE
{
        time-interval           [0]     INTEGER (1..80),
                                        -- relative to current time stamp
                                        -- units = 15 minutes
                                        -- range =15 minutes to 20 hours
        number-of-way-points    [1]     INTEGER (1..128)
}

GroundSpeedChange ::= INTEGER (0..300)
        -- units =Knots
        -- range = 0 to 300 knots

Ias ::= INTEGER(0..1100)
        -- units =knots
        -- range =0 to 1100 knots

LateralChange ::= INTEGER (0..2000)
        -- units = 0.1 Nautical miles
        -- range= 0 to 200 Nautical miles

Mach ::= INTEGER (500..4000)
        -- units = Mach 0.001
        -- range =0.5 Mach to 4 Mach

ShortTermIntentModulus ::= SEQUENCE
```

```
{
        intent-modulus                  Modulus,
        intent-projection-time          ProjectionTime
}

ProjectionTime ::= INTEGER (1..240)
        -- units = minutes relative to current time stamp
        -- range = 1 minute to 4 hours

ReportingInterval ::= CHOICE
{
        seconds-scale           [0]     INTEGER (1..59),
                                        -- units = seconds
                                        -- range = 1 second to 59 seconds
        minutes-scale           [1]     INTEGER (1..120)
                                        -- units = minutes
                                        -- range = 1 minute to 2 hours
}

VerticalRateChange ::= INTEGER (-3000..3000)
        -- units = 10 feet per minute
        -- range =-30 000 to +30 000 feet per minute

-- ------------------------------------------------------------------------------------------------------------
-- Miscellaneous components
-- ------------------------------------------------------------------------------------------------------------

Reason ::= CHOICE
{
        aDS-service-unavailable         [0]     NULL,
        undefined                       [1]     NULL,
        -- the undefined value should not be used
        maximum-capacity-exceeded       [2]     GroundSystemsUsingService,
        undefined-reason                [3]     NULL,
        ...
}

GroundSystemsUsingService ::= SEQUENCE OF IA5String (SIZE(4..8))
-- contains a sequence of ICAO facility designations

RequestType ::= ENUMERATED
{
        event-contract                  (0),
        periodic-contract               (1),
        demand-contract                 (2),
        cancel-event-contract           (3),
```

```
        cancel-periodic-contract            (4),
        modify-emergency-contract           (5),
        cancel-all-contracts                (6),
        ...
}

EventTypeContracted ::= ENUMERATED
{
        lateral-deviation-change            (0),
        vertical-rate-change                (1),
        level-threshold                     (2),
        way-point-change                    (3),
        air-speed-change                    (4),
        ground-speed-change                 (5),
        heading-change                      (6),
        extended-projected-profile-change   (7),
        fom-change                          (8),
        track-angle-change                  (9),
        level-change                        (10),
        ...
}

EventTypeReported ::= ENUMERATED
{
        lateral-deviation-change            (0),
        vertical-rate-change                (1),
        level-threshold                     (2),
        way-point-change                    (3),
        air-speed-change                    (4),
        ground-speed-change                 (5),
        heading-change                      (6),
        extended-projected-profile-change   (7),
        fom-change                          (8),
        track-angle-change                  (9),
        level-change                        (10),
        baseline                            (11),
        ability-to-detect-events-impaired   (12),
        ...
}

Modulus ::= INTEGER (1..255)

ReportType ::= ENUMERATED
{
        aircraft-address                    (0),
        projected-profile                   (1),
```

```
        ground-vector                   (2),
        air-vector                      (3),
        weather                         (4),
        short-term-intent               (5),
        extended-projected-profile      (6),
        ...
}
ReportTypeAndPeriod ::= ENUMERATED
{
        aircraft-address                (0),
        projected-profile               (1),
        ground-vector                   (2),
        air-vector                      (3),
        weather                         (4),
        short-term-intent               (5),
        extended-projected-profile      (6),
        reporting-rate                  (7),
        ...
}

-- ------------------------------------------------------------------------------------------------------------
-- Common components
-- ------------------------------------------------------------------------------------------------------------

Eta ::=Time

Position ::= SEQUENCE
{
        latitude            Latitude,
        longitude                   Longitude,
        level                       Level
}

Latitude ::= SEQUENCE
{
        sign                            Sign,
        degrees                 INTEGER (0..90),
                                        -- units = degrees
                                        -- range = 0 degrees to 90 degrees
        minutes                         INTEGER (0..59),
                                        -- units = minutes
                                        -- range = 0 minutes to 59 minutes
        tenth-seconds                   INTEGER (0..599)
                                        -- units = 0.1 seconds
                                        -- range = 0 seconds to 59.9 seconds
```

```
}

Longitude ::= SEQUENCE
{
        sign                            Sign,
        degrees                 INTEGER (0..180),
                                        -- units = degrees
                                        -- range = 0 degrees to 180 degrees
        minutes                         INTEGER (0..59),
                                        -- units = minutes
                                        -- range = 0 minutes to 59 minutes
        tenth-seconds                   INTEGER (0..599)
                                        -- units = 0.1 seconds
                                        -- range = 0 seconds to 59.9 seconds

}

Sign ::= ENUMERATED
{
        plus                            (0),
        minus                           (1)
}

Level    ::= INTEGER(-75..10000)
        -- units = 10 feet
        — range = -750 to 100 000 feet

DateTimeGroup ::= SEQUENCE
{
        date        Date,
        time        Time
}

Date ::= SEQUENCE
{
        year                            Year,
        month                           Month,
        day                             Day
}

Year ::= INTEGER (1996..2095)
        -- unit = year
        -- range = 1996 to 2095

Month ::= INTEGER (1..12)
        -- unit = month
```

```
        -- range = January to December

Day ::= INTEGER (1..31)
        -- unit = day
        -- range = 1 to 31

Time ::= SEQUENCE
{
        timeHours               [0]     TimeHours,
        timeMinutes             [1]     TimeMinutes,
        timeSeconds             [2]     TimeSeconds         OPTIONAL
}

TimeHours ::= INTEGER (0..23)
        -- units = hours
        -- range = midnight to 23.00 (11 PM)

TimeMinutes ::= INTEGER (0..59)
        -- units = minutes
        -- range = 0 minutes to 59 minutes

TimeSeconds ::= INTEGER (0..59)
        -- units = seconds
        -- range = 0 seconds to 59 seconds

END     -- of ADSMessageSetVersion1
```

2.2.1.5 Protocol Definition

2.2.1.5.1 Sequence Rules

2.2.1.5.1.1 Only the sequence of primitives illustrated in figures 2.2.1.5-1 to 2.2.1.5-35 shall be permitted.

Note 1.— The following figures define the valid sequences of primitives that are possible to be invoked during the operation of the ADS application. 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.— Abort primitive may interrupt and terminate any of the normal message sequences outlined below.

Note 3.— Primitives are processed in the order in which they are received (see 4.4.4.1.2.4).

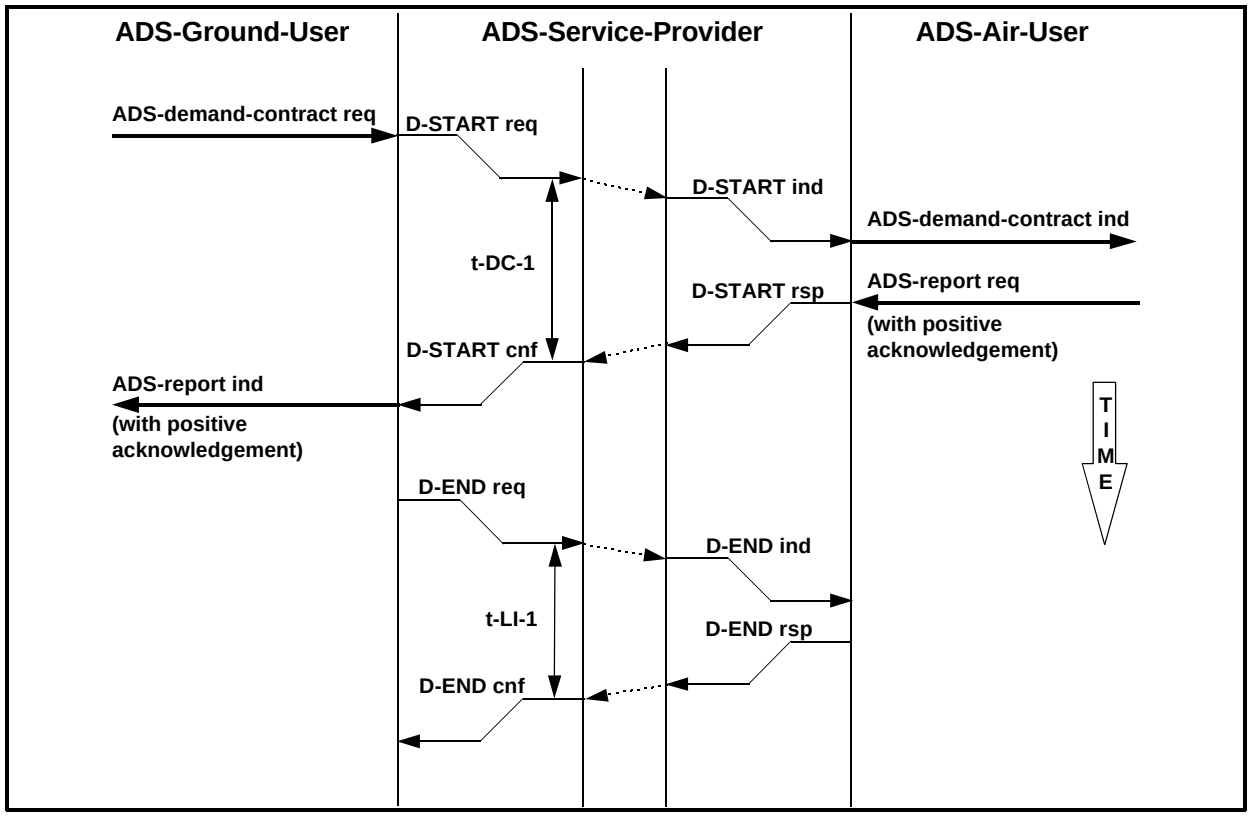

Figure 2.2.5.1-1: Use of demand contract with no dialogue existing

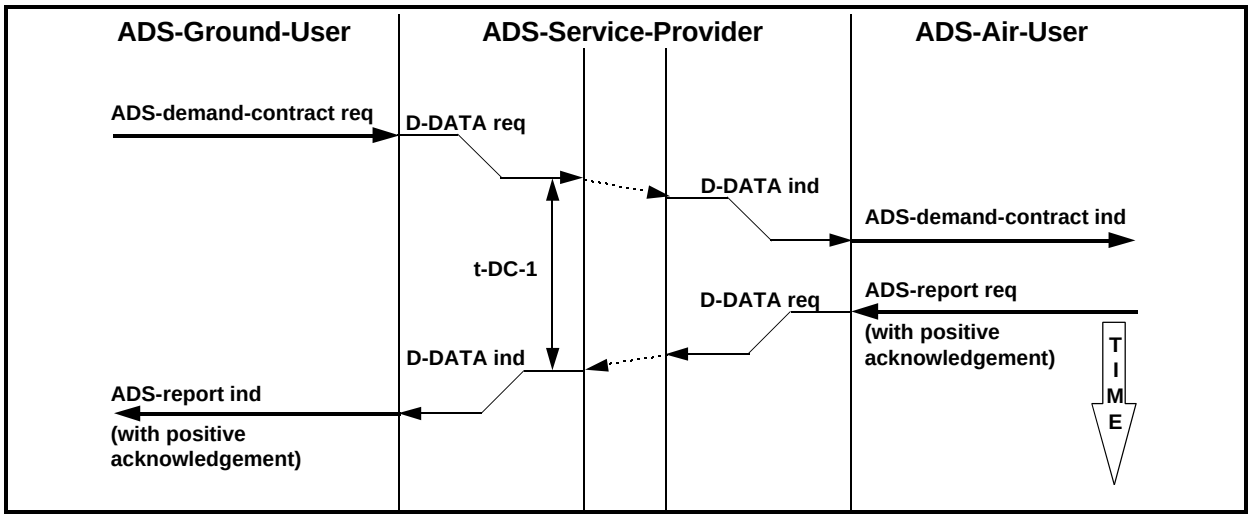

Figure 2.2.1.5-2: Use of demand contract with dialogue existing

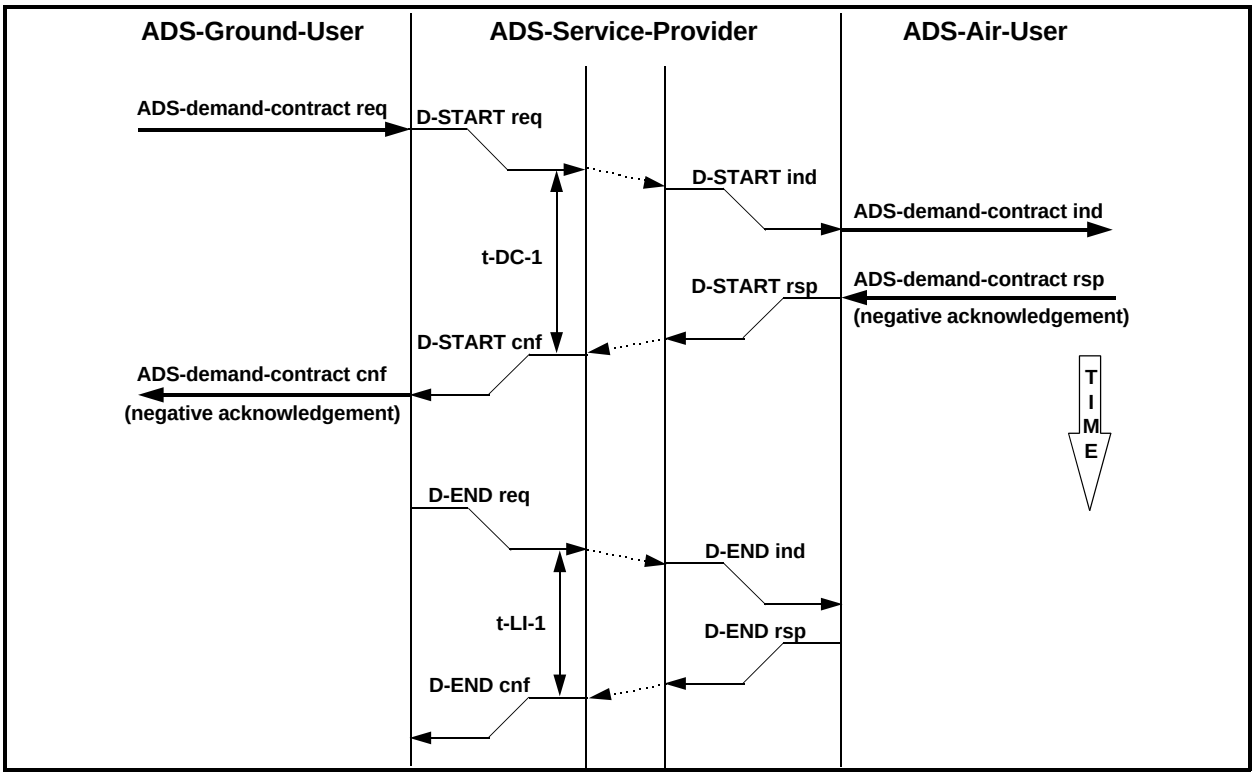

Figure 2.2.1.5-3: Use of demand contract with no dialogue existing

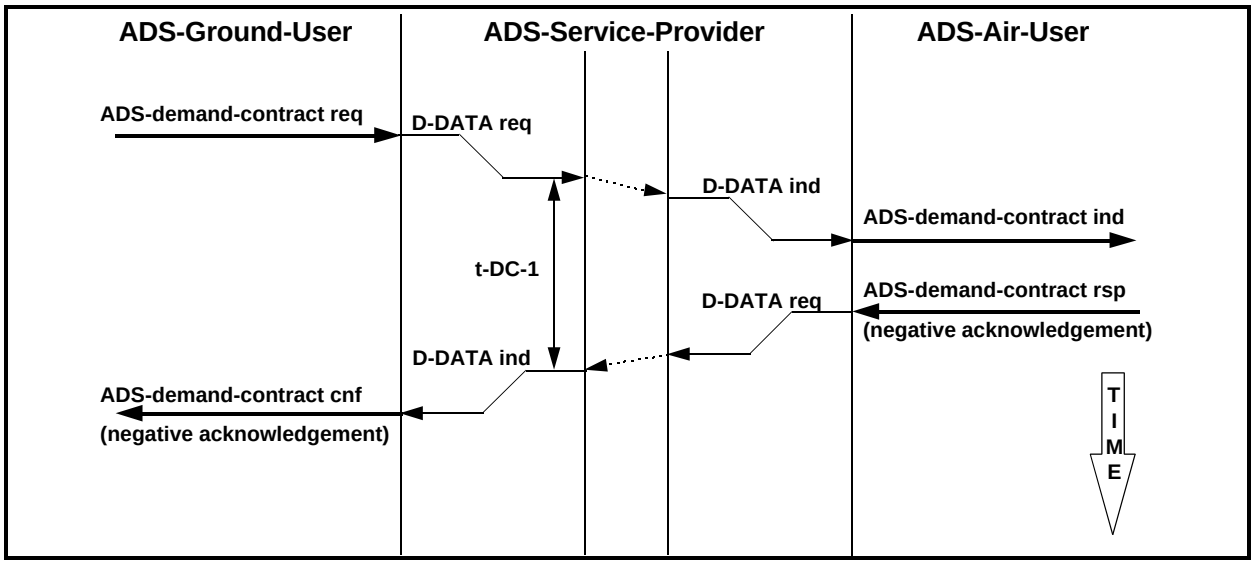

Figure 2.2.1.5-4: Use of demand contract with dialogue existing - with negative acknowledgement

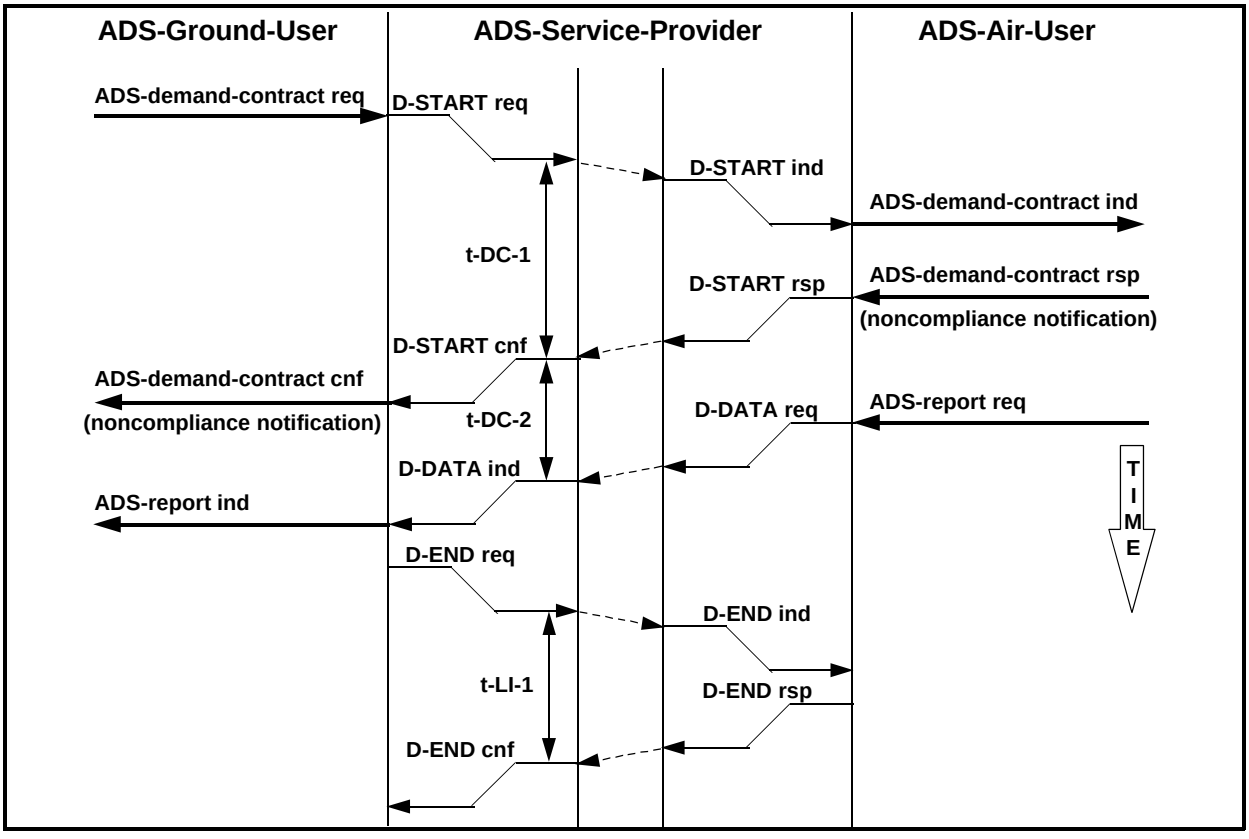

Figure 2.2.1.5-5: Use of demand contract with no dialogue existing - with noncompliance notification

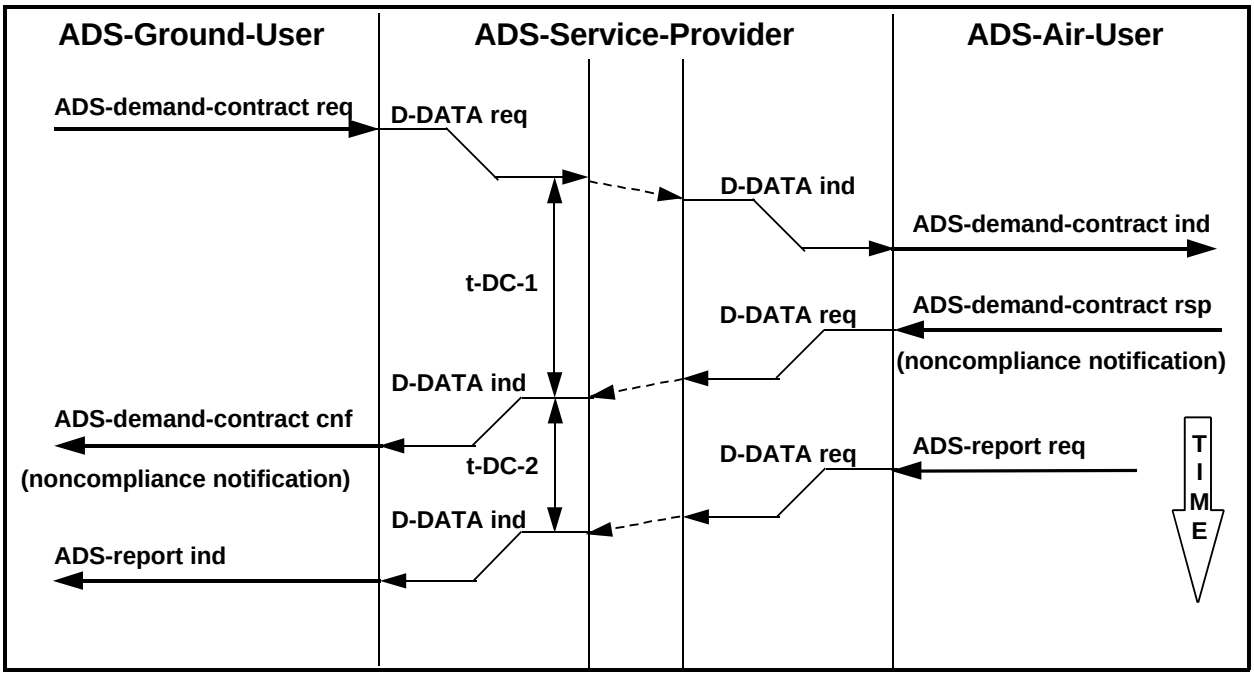

Figure 2.2.1.5-6: Use of demand contract with dialogue existing - with noncompliance notification

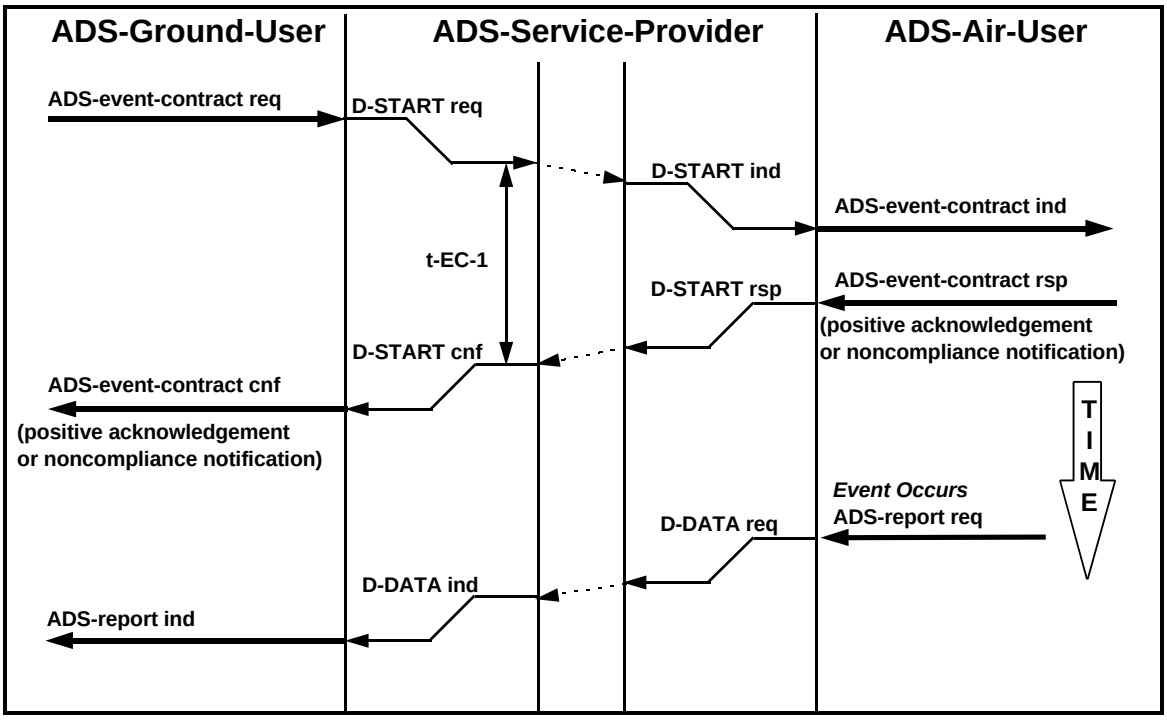

Figure 2.2.1.5-7: Use of event contract with no dialogue existing - with positive acknowledgement or noncompliance notification

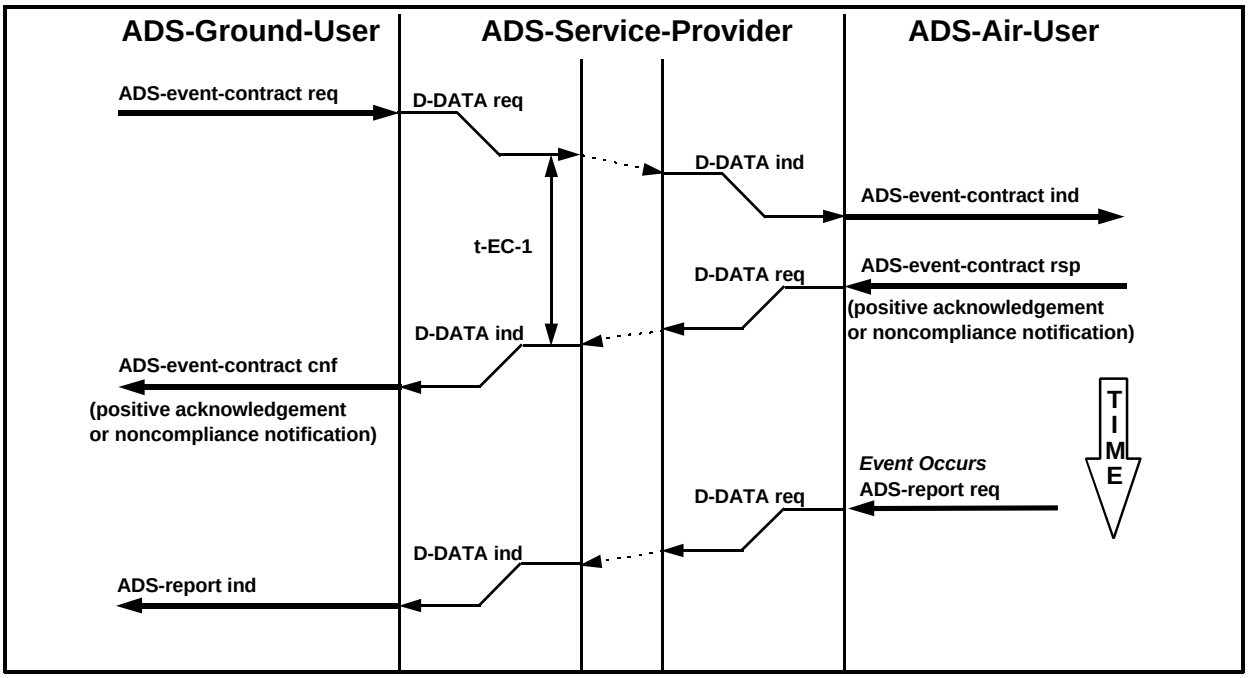

Figure 2.2.1.5-8: Use of event contract with dialogue existing - with positive acknowledgement or noncompliance notification

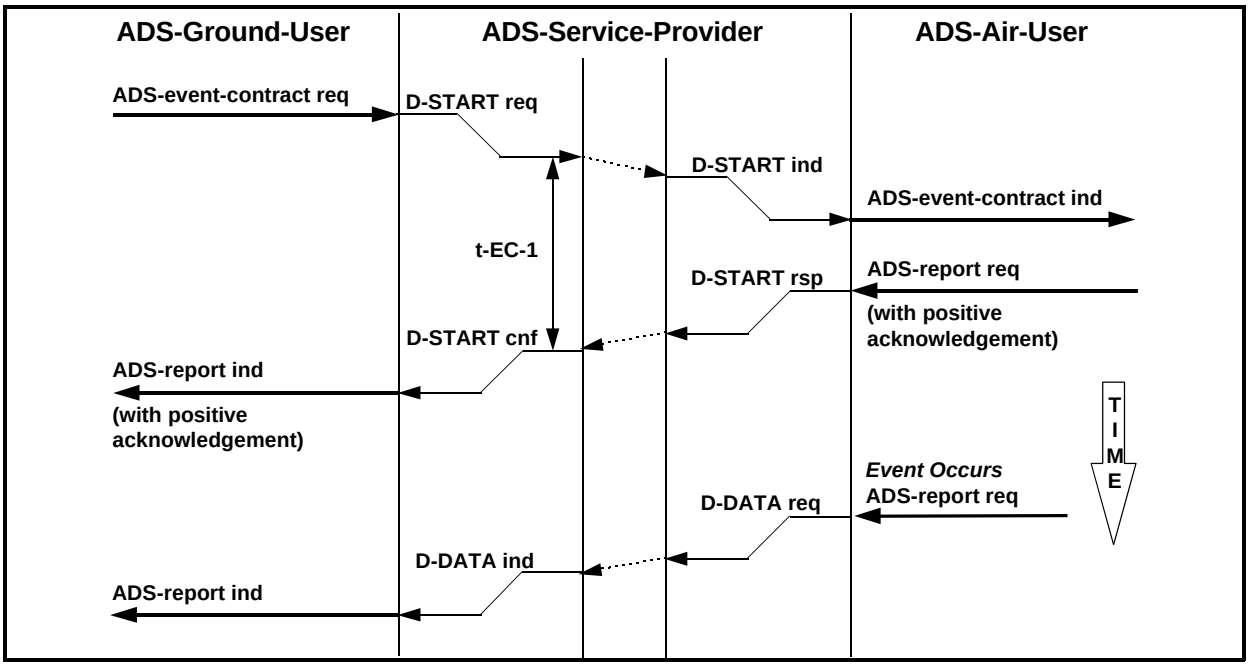

Figure 2.2.1.5-9: Use of event contract with no dialogue existing

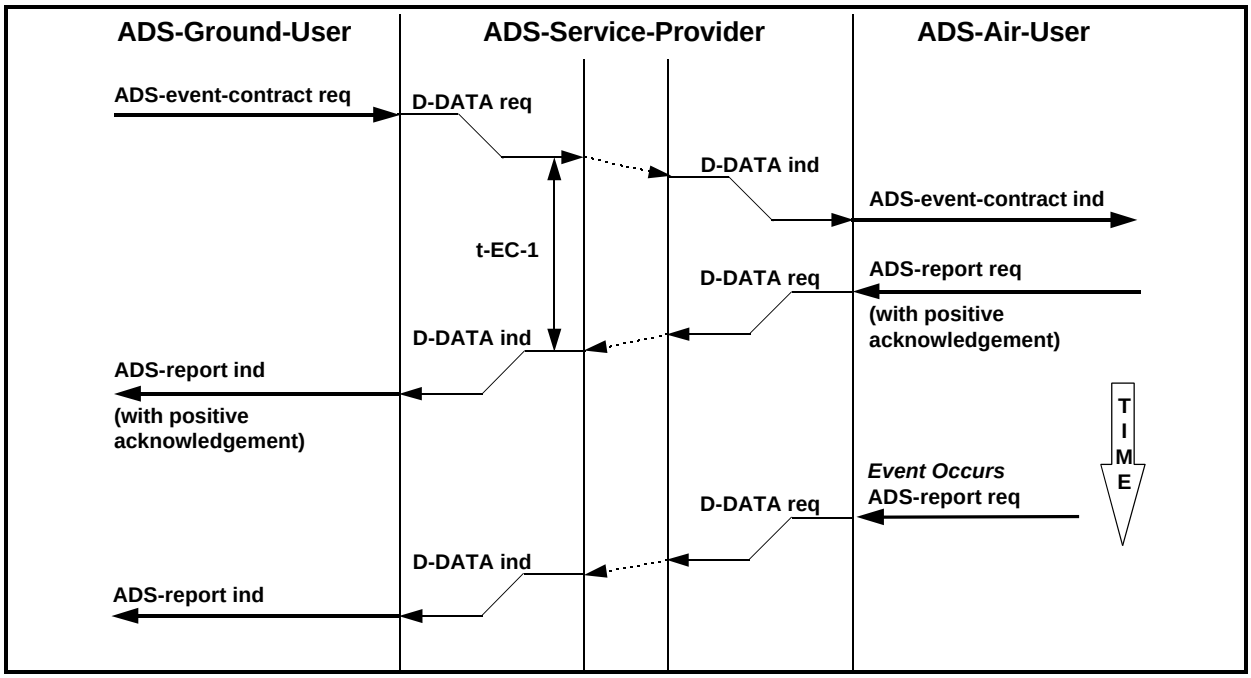

Figure 2.2.1.5-10: Use of event contract with dialogue existing

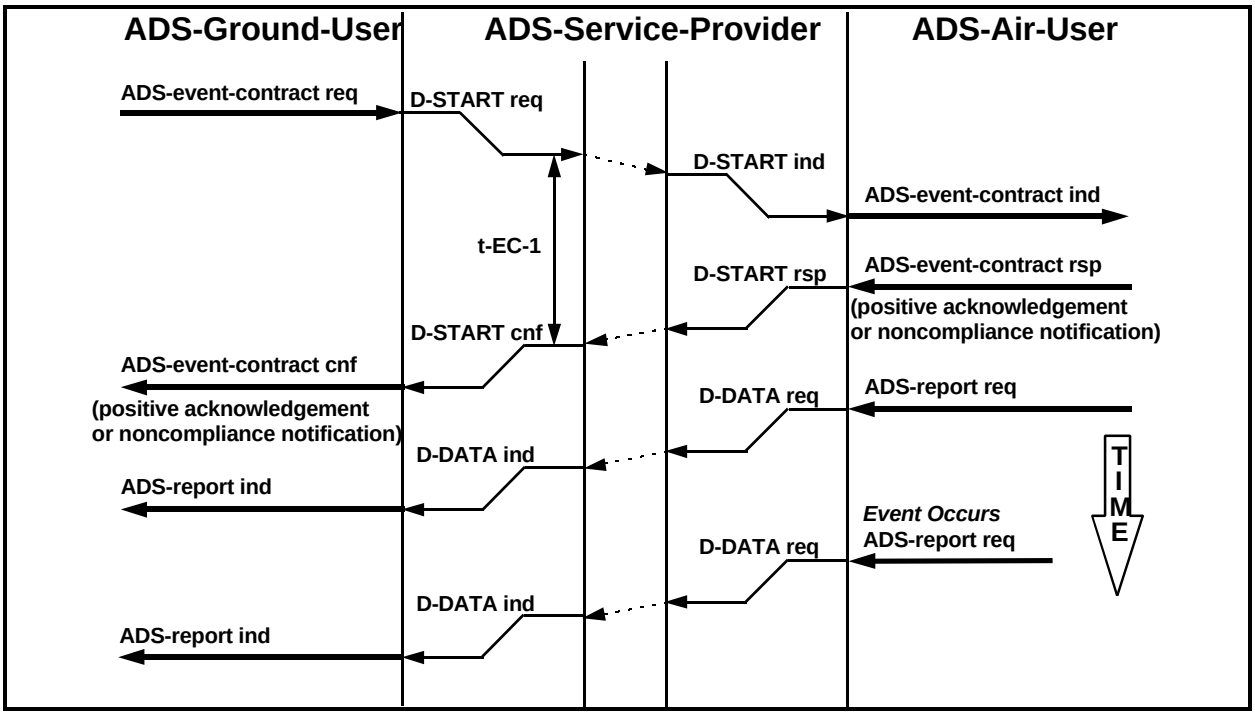

Figure 2.2.1.5-11: Use of event contract with no dialogue existing - with positive acknowledgement or noncompliance notification and immediate report

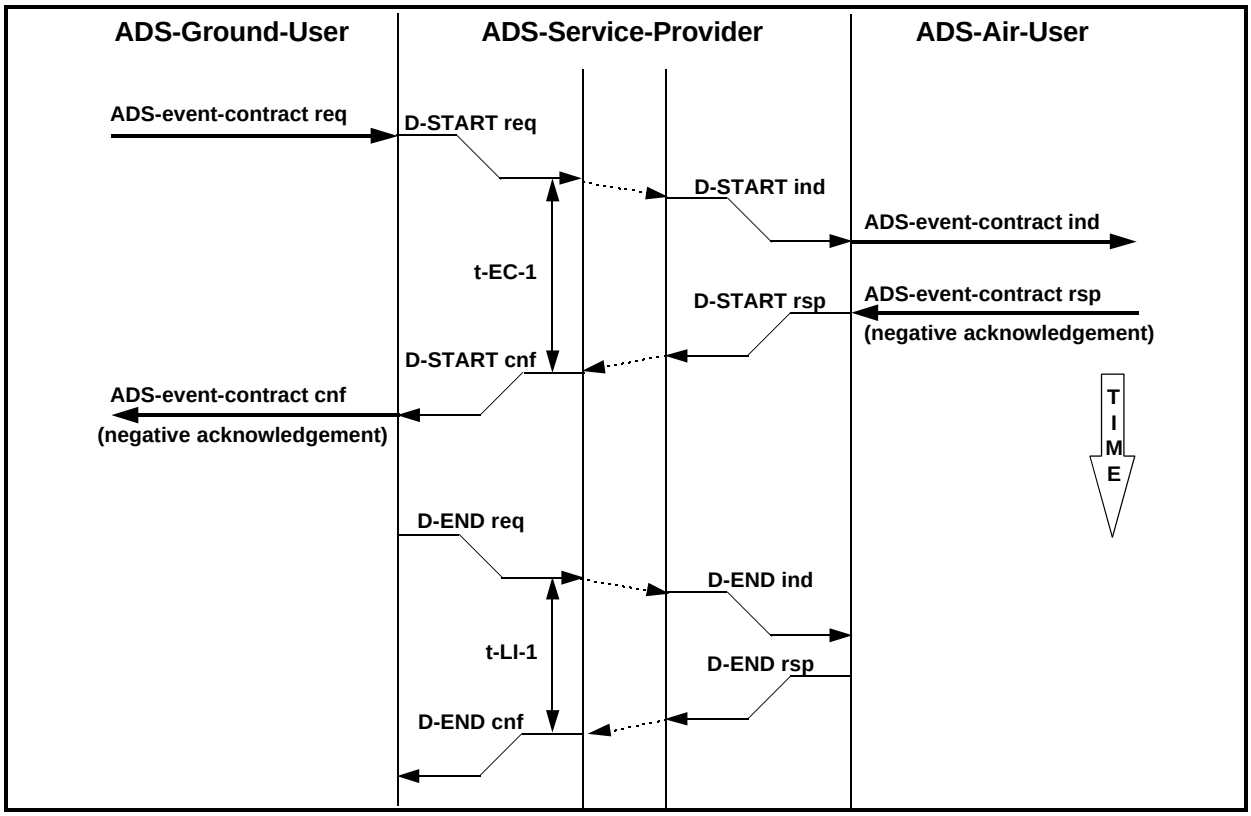

Figure 2.2.1.5-12: Use of event contract with no dialogue existing - with negative acknowledgement

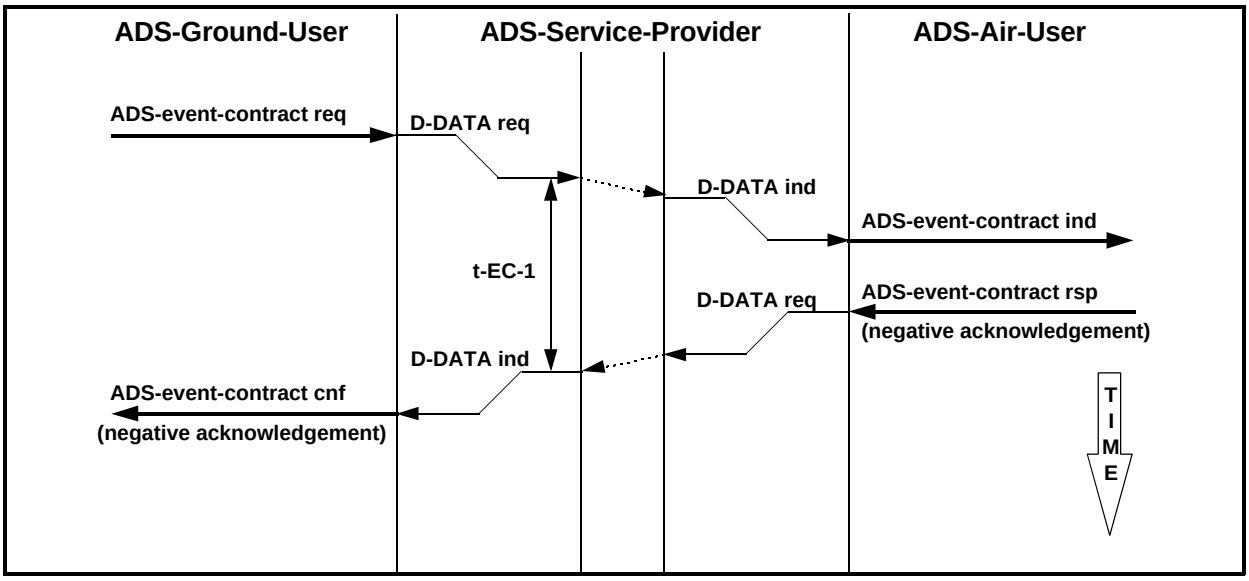

Figure 2.2.1.5-13: Use of event contract with dialogue existing - with negative acknowledgement

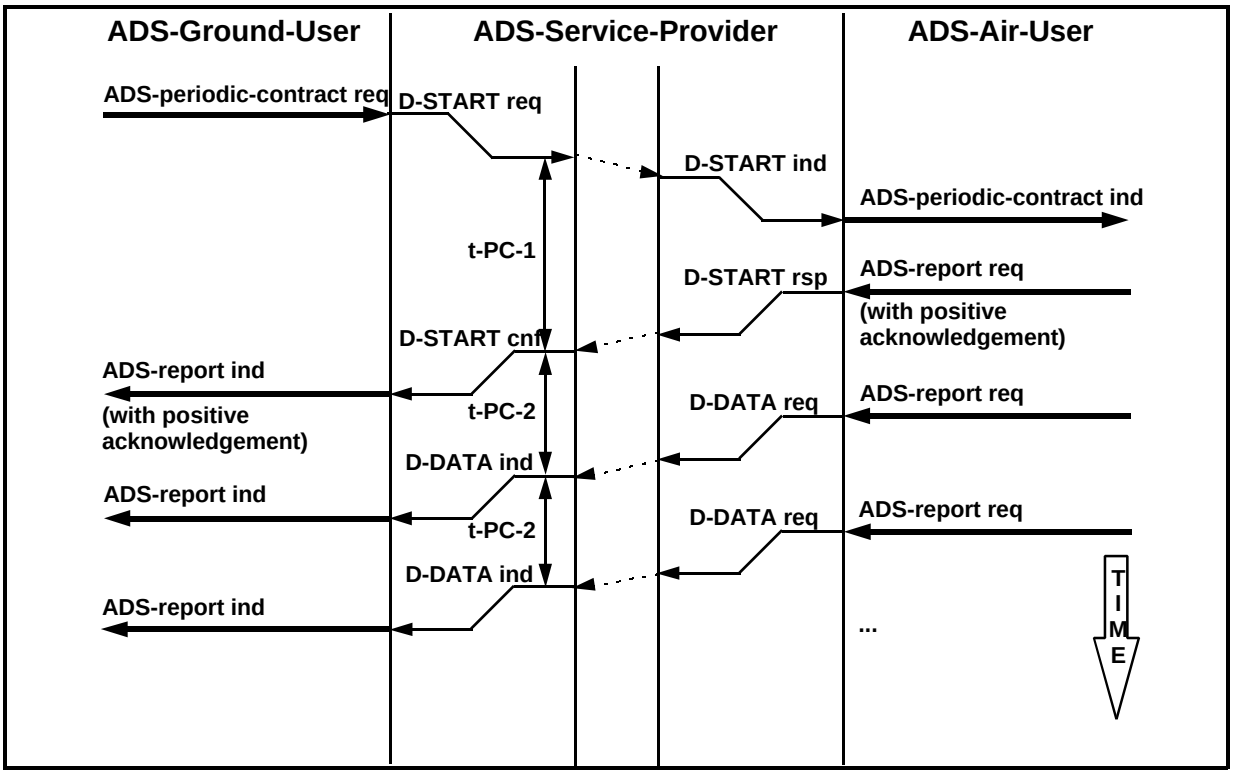

Figure 2.2.1.5-14: Use of periodic contract with no dialogue existing

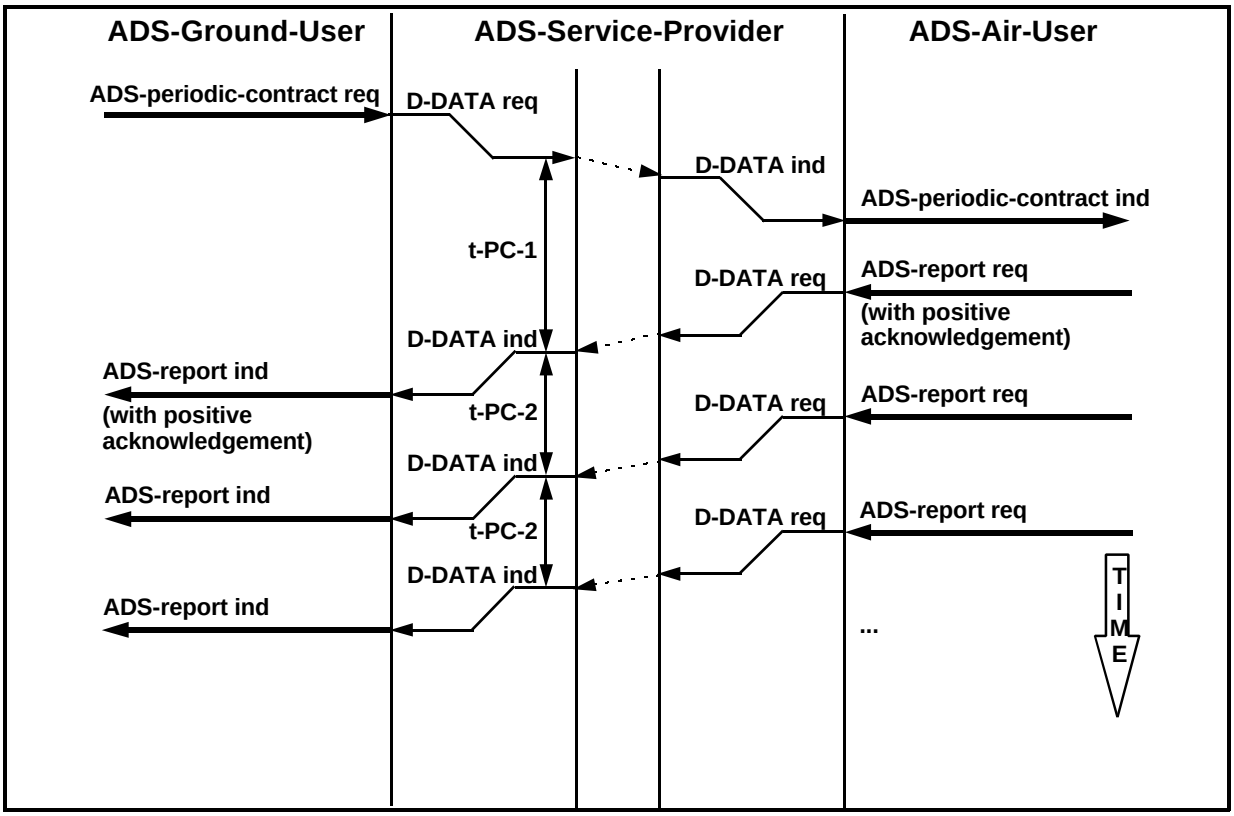

Figure 2.2.1.5-15: Use of periodic contract with a dialogue existing

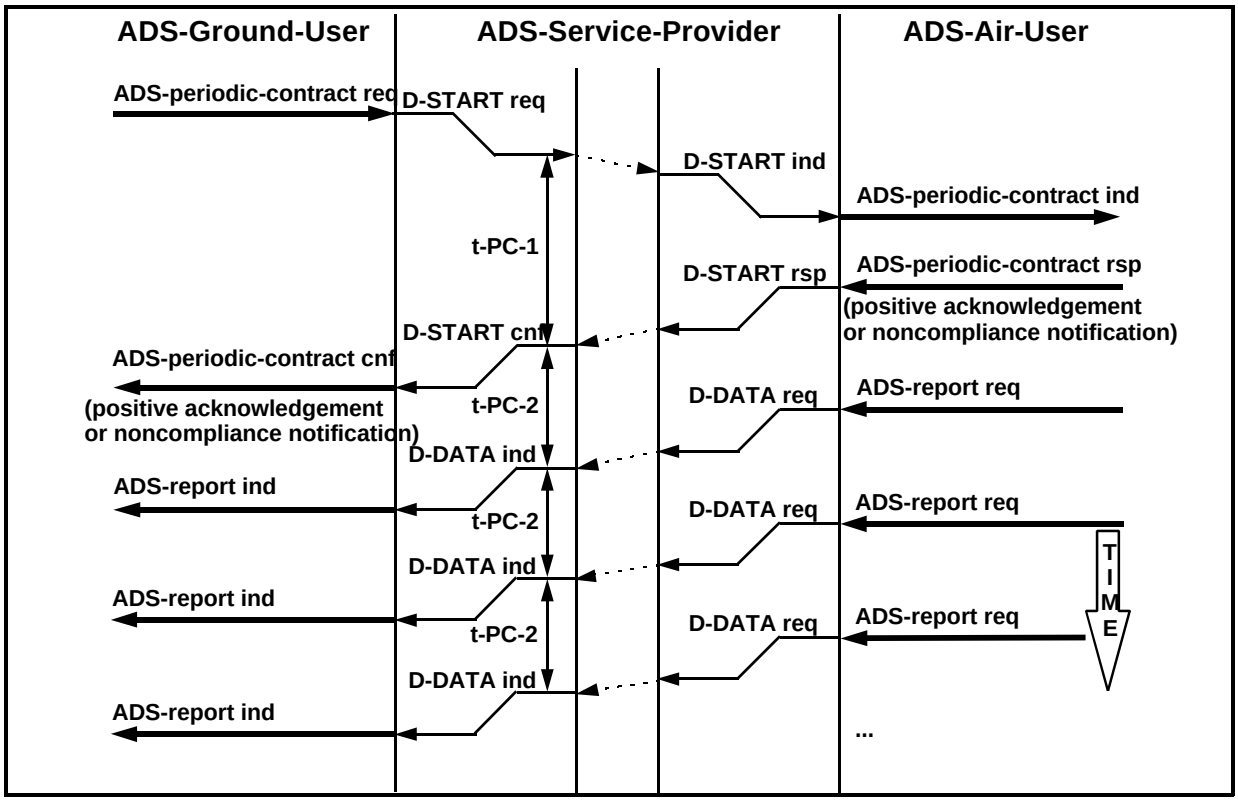

Figure 2.2.1.5-16: Use of periodic contract with no dialogue existing with positive acknowledgement or noncompliance notification

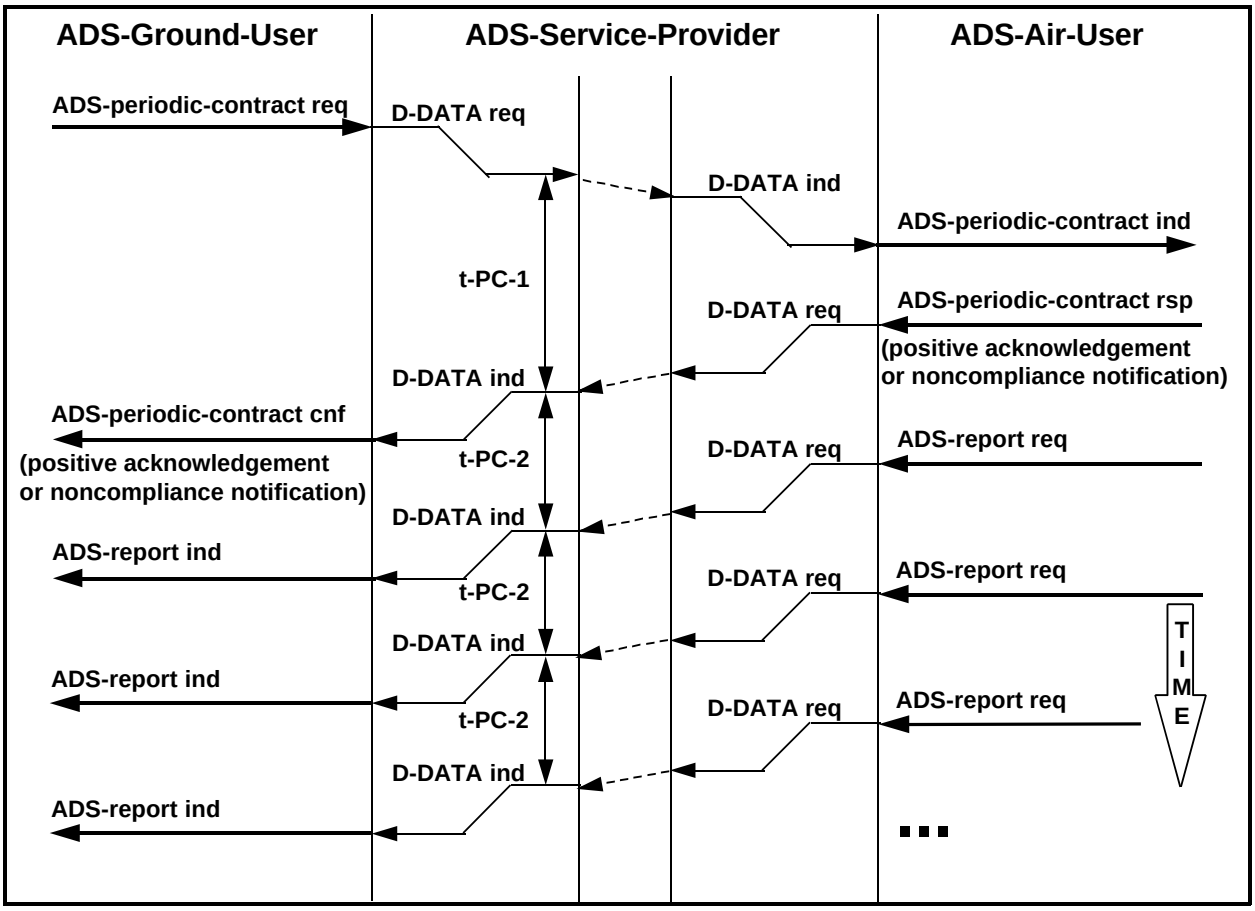

Figure 2.2.1.5-17: Use of periodic contract with dialogue existing with positive acknowledgement or noncompliance notification

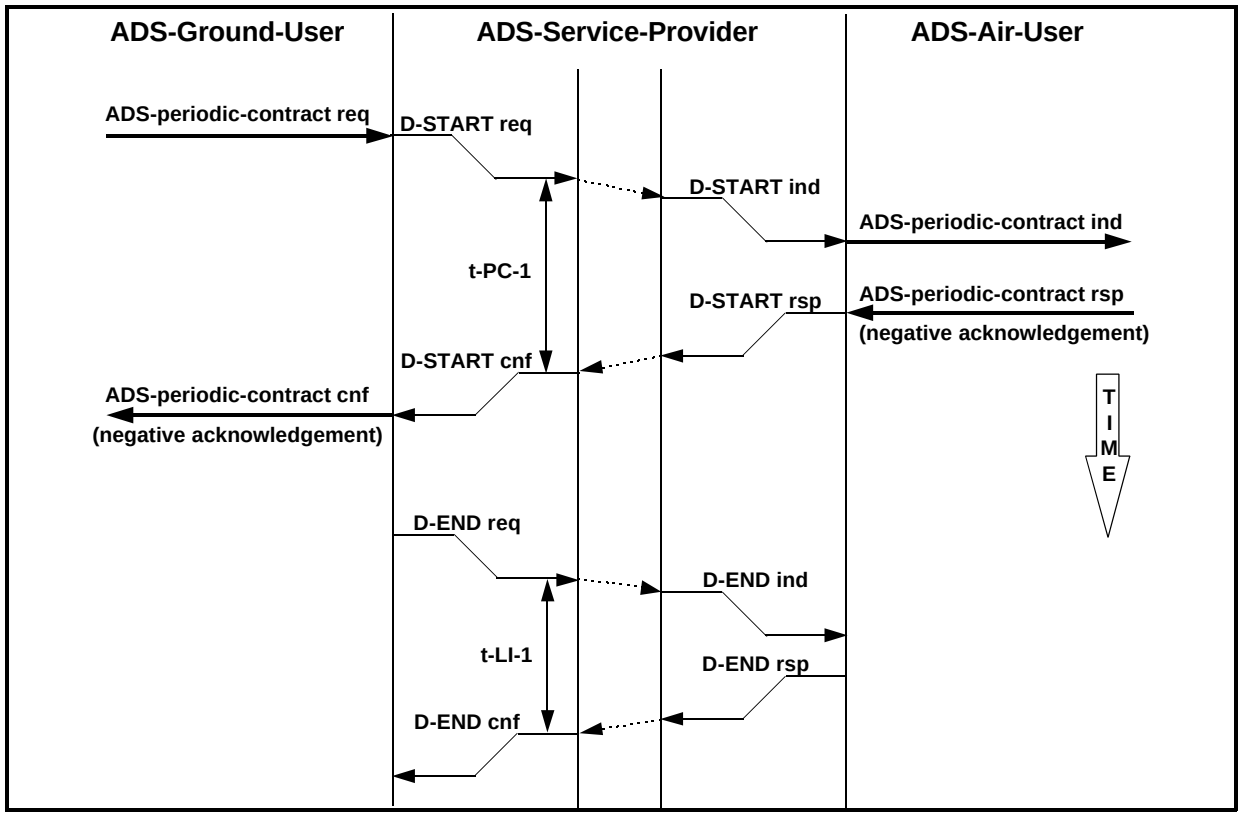

Figure 2.2.1.5-18: Use of periodic contract with no dialogue existing with negative acknowledgement

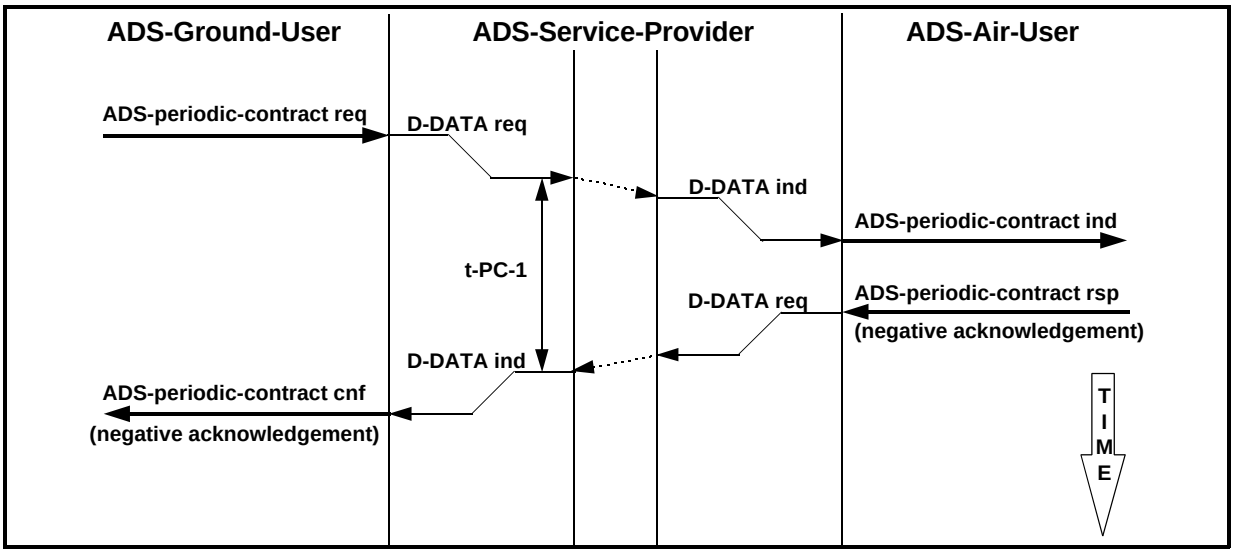

Figure 2.2.1.5-19: Use of periodic contract with a dialogue existing with negative acknowledgement

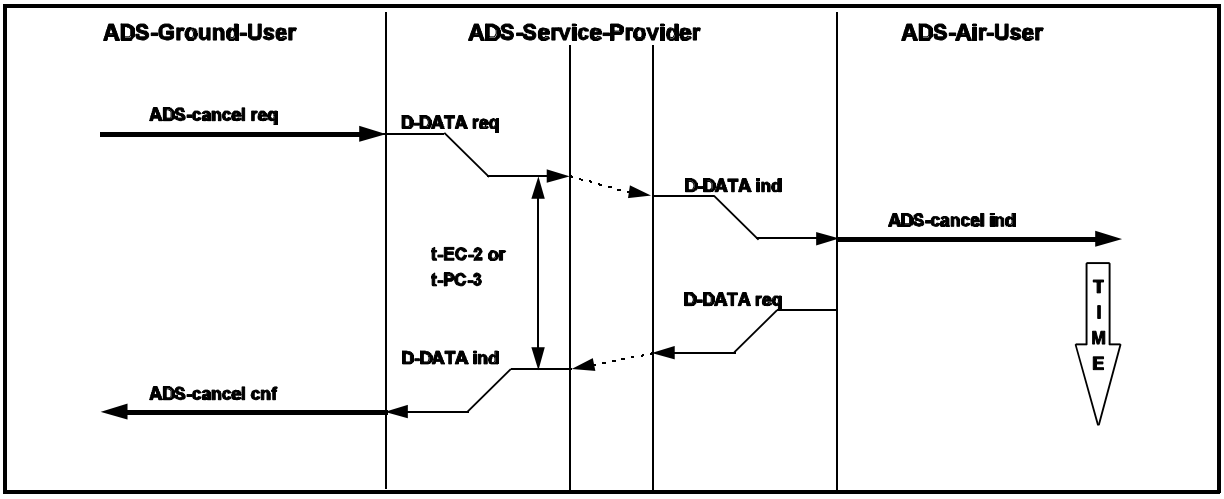

Figure 2.2.1.5-20: Use of ADS cancel contract service

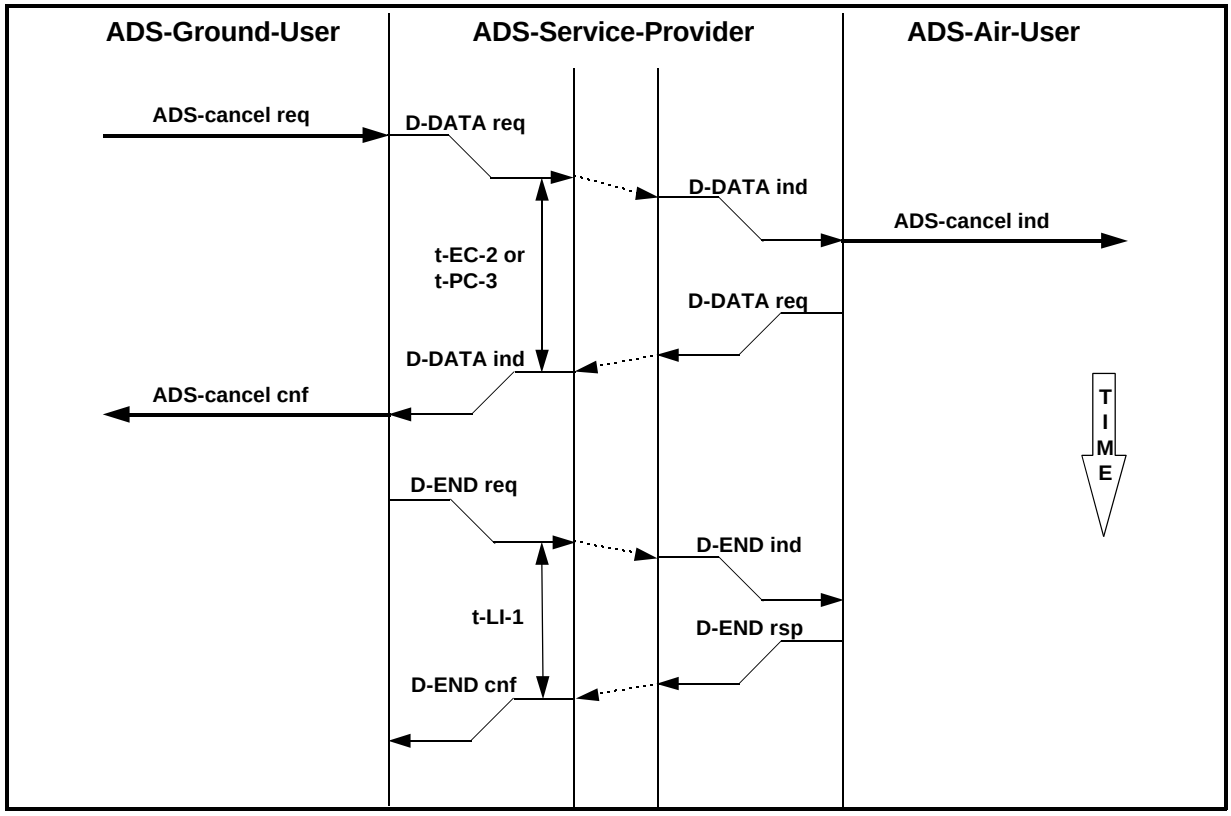

Figure 2.2.1.5-21: Use of ADS cancel contract service with only one contract

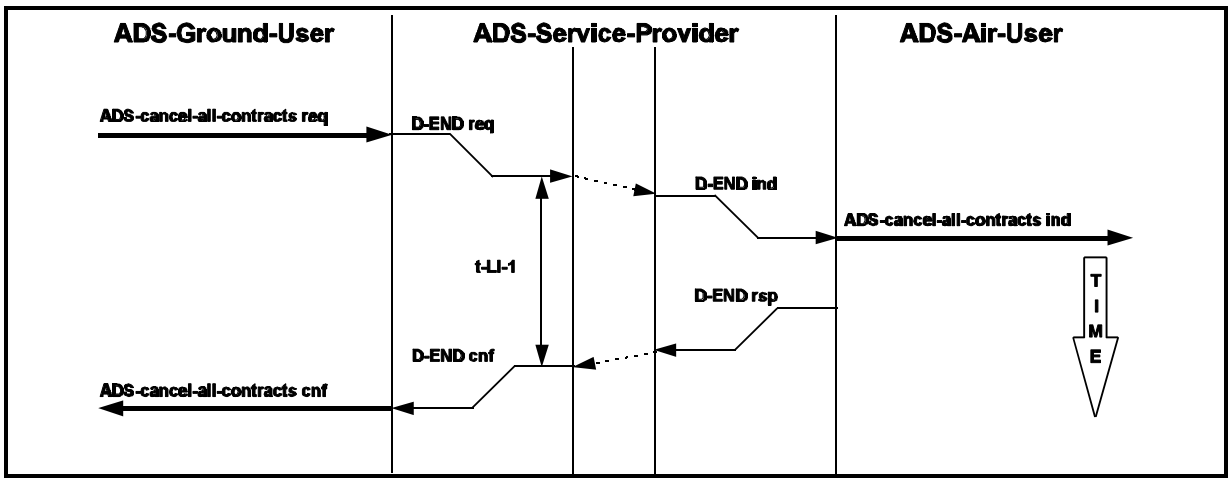

Figure 2.2.1.5-22: Use of ADS cancel all contracts service

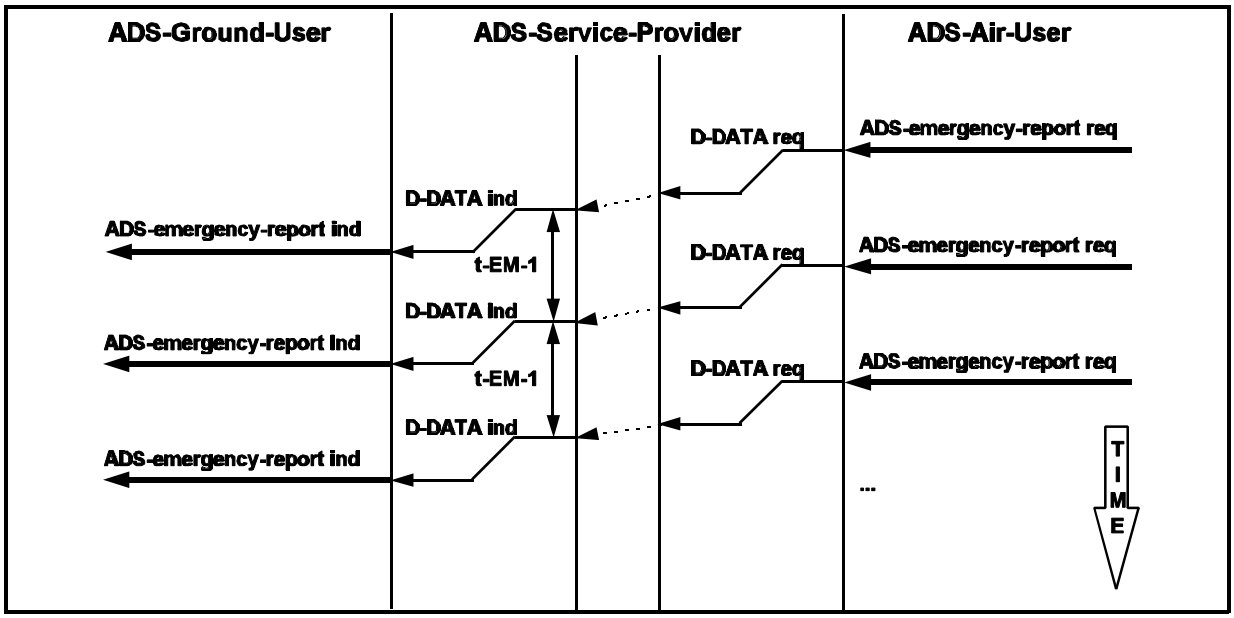

Figure 2.2.1.5-23: Use of emergency report service

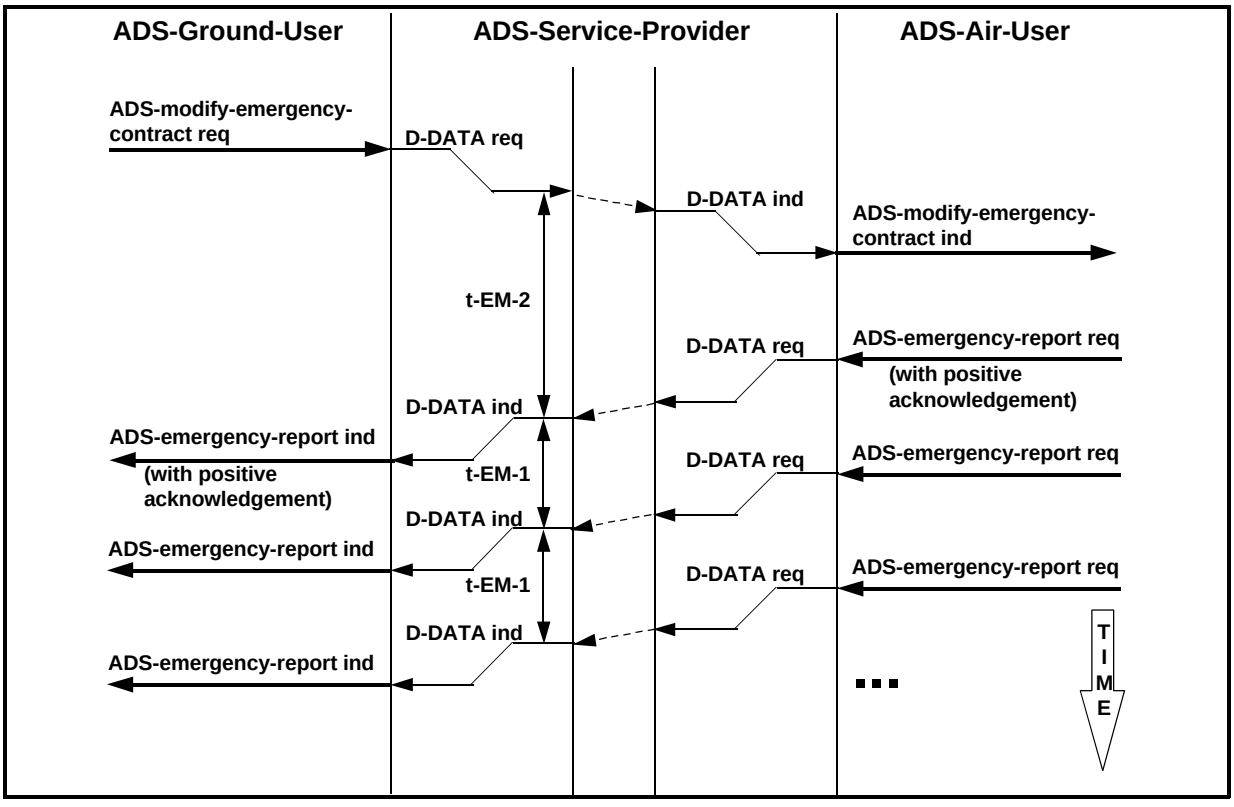

Figure 2.2.1.5-24: Modification of emergency contract

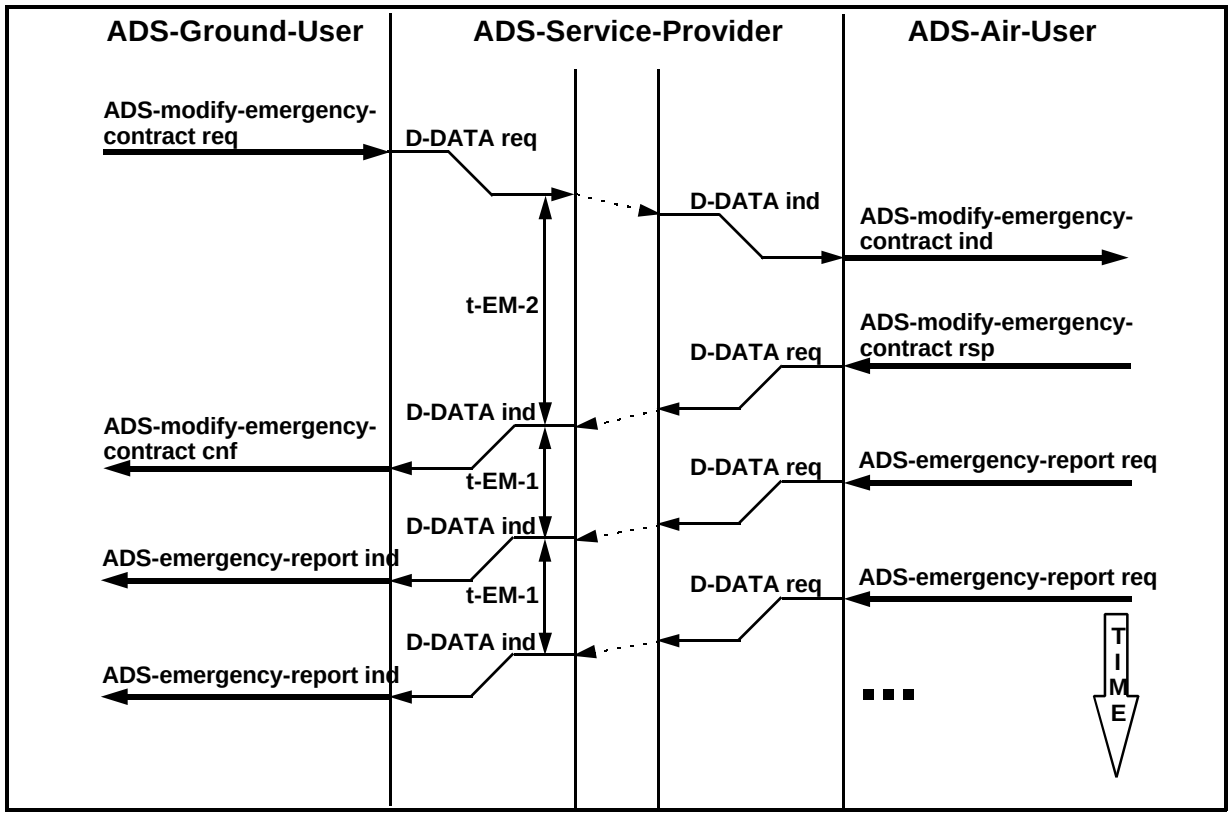

Figure 2.2.1.5.25: Modification of emergency contract rejected

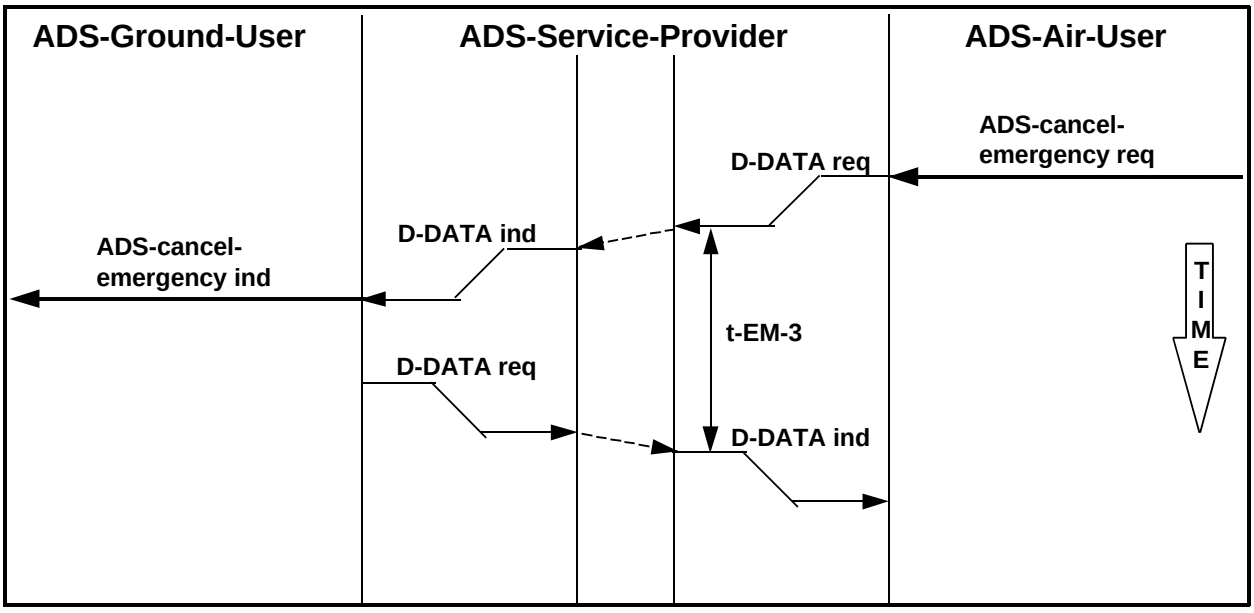

Figure 2.2.1.5-26: Cancellation of emergency contract

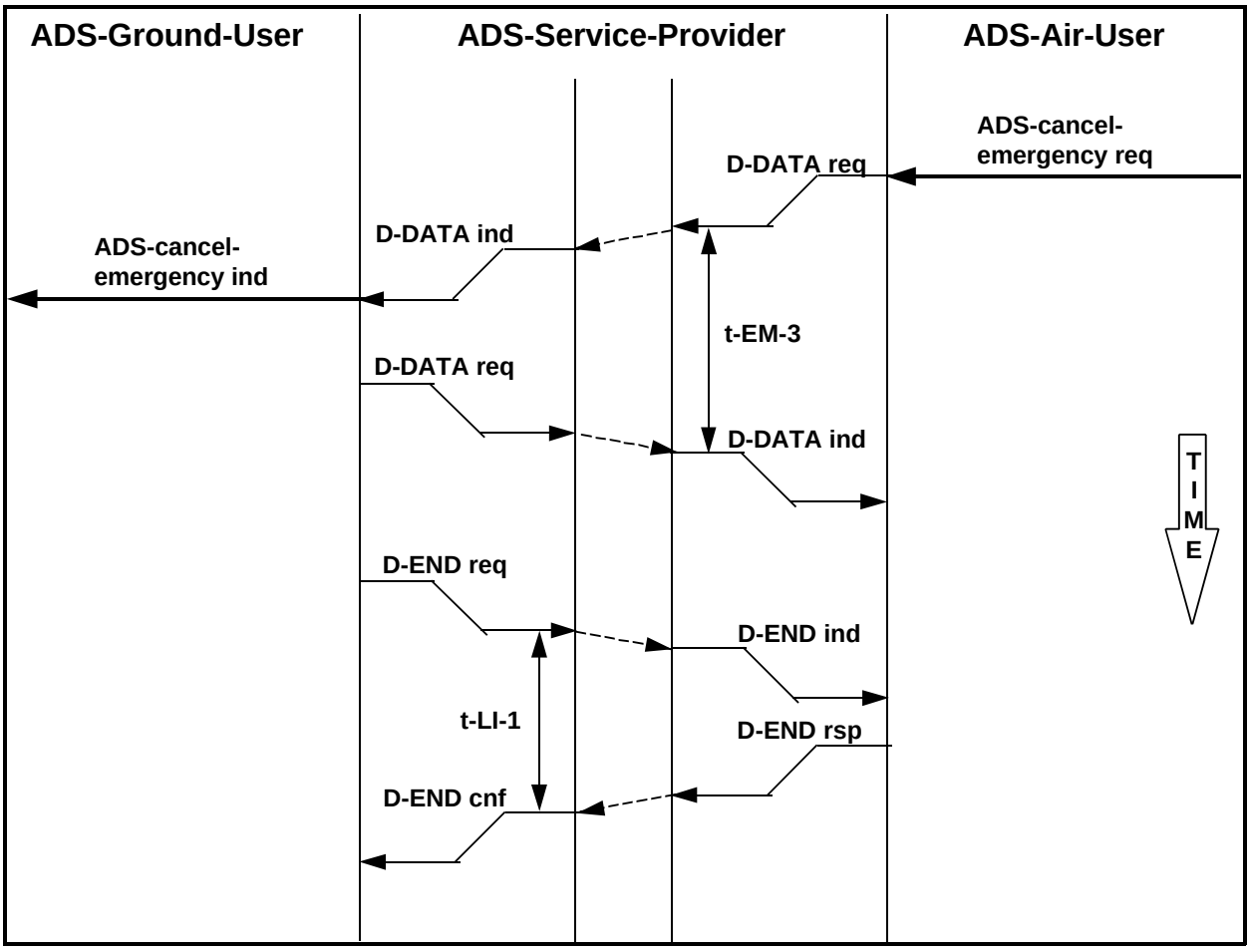

Figure 2.2.1.5-27: Cancellation of emergency contract with no other contracts in place

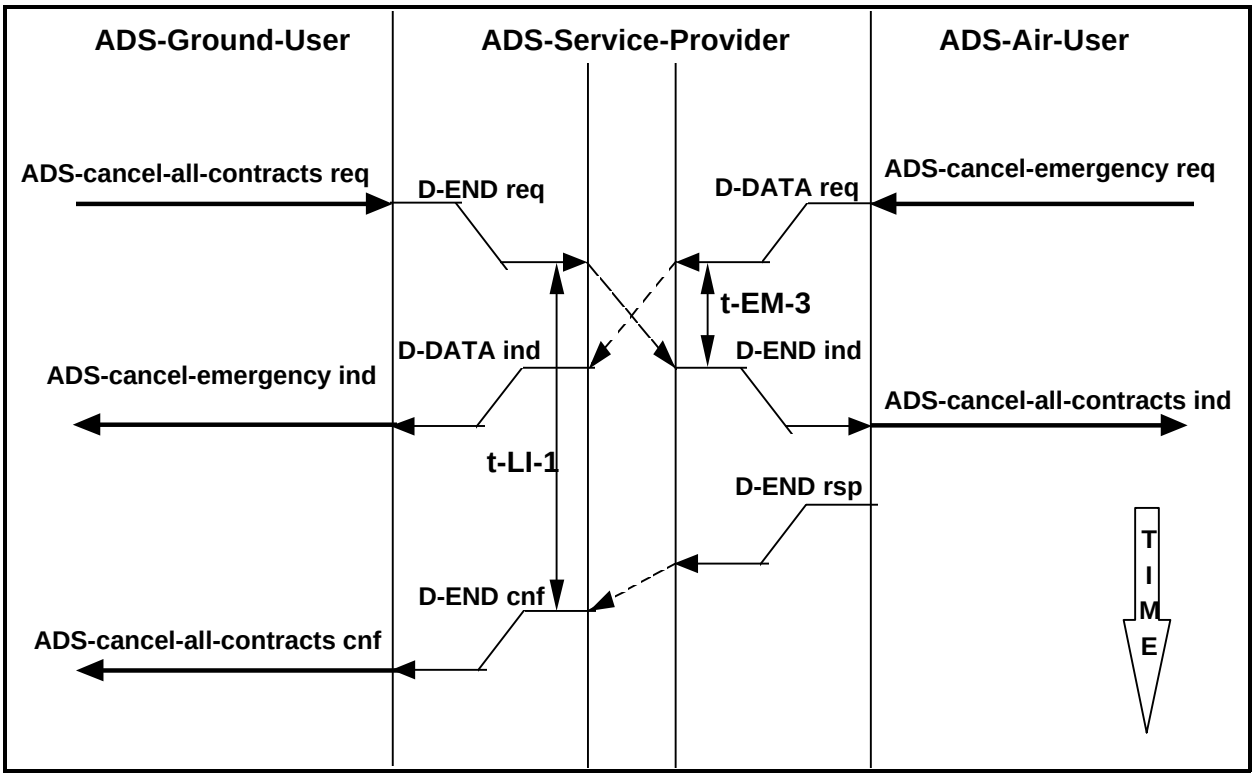

Figure 2.2.1.5-28: Crossed air emergency cancellation and cancel all contracts

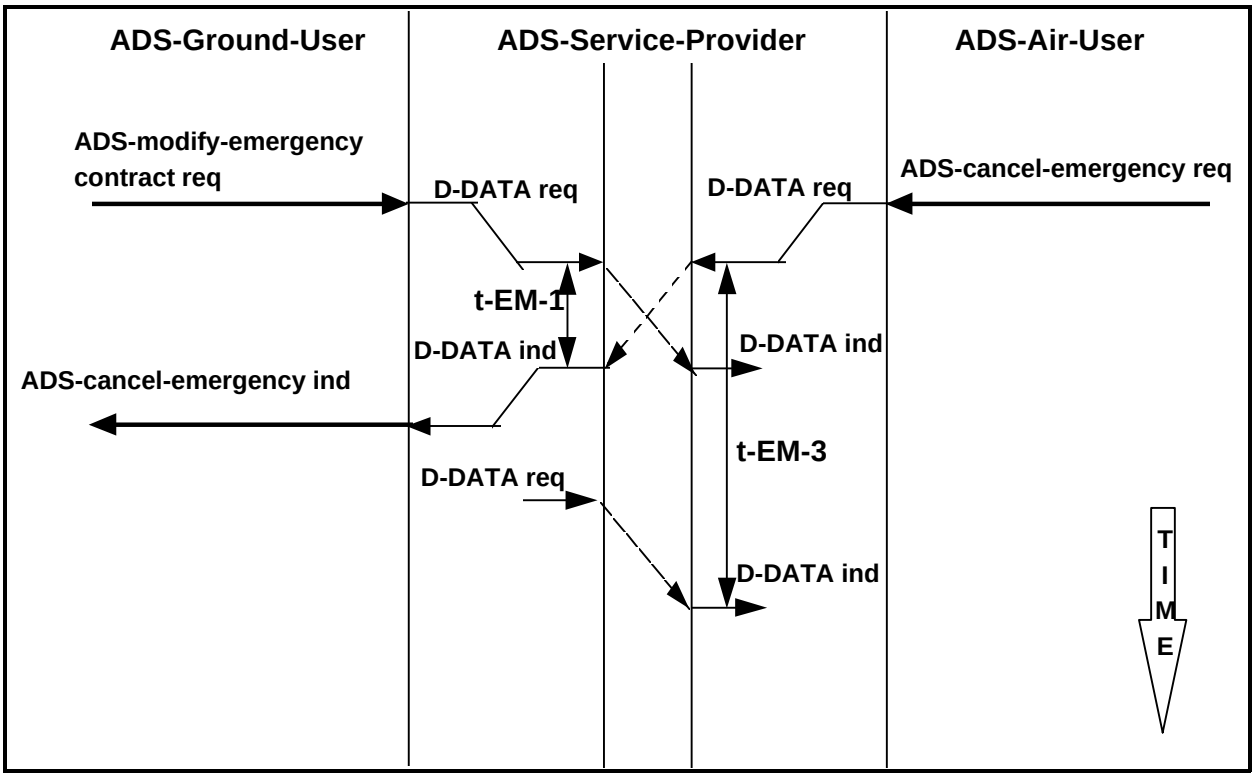

Figure 2.2.1.5-29: Crossed air emergency cancellation and modification of emergency contract with other contracts in place

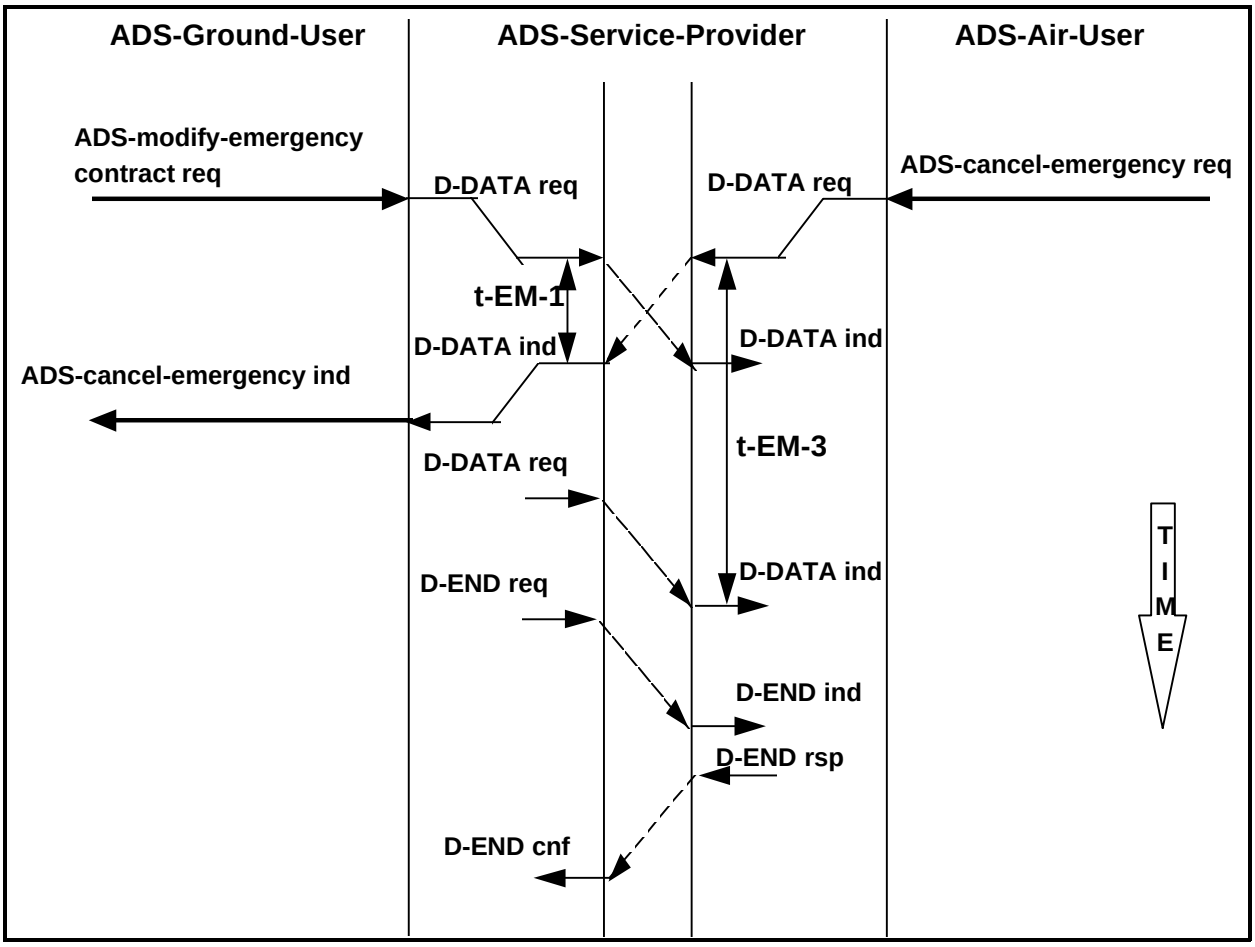

Figure 2.2.1.5-30: Crossed air emergency cancellation and modification of emergency contract with no other contracts in place

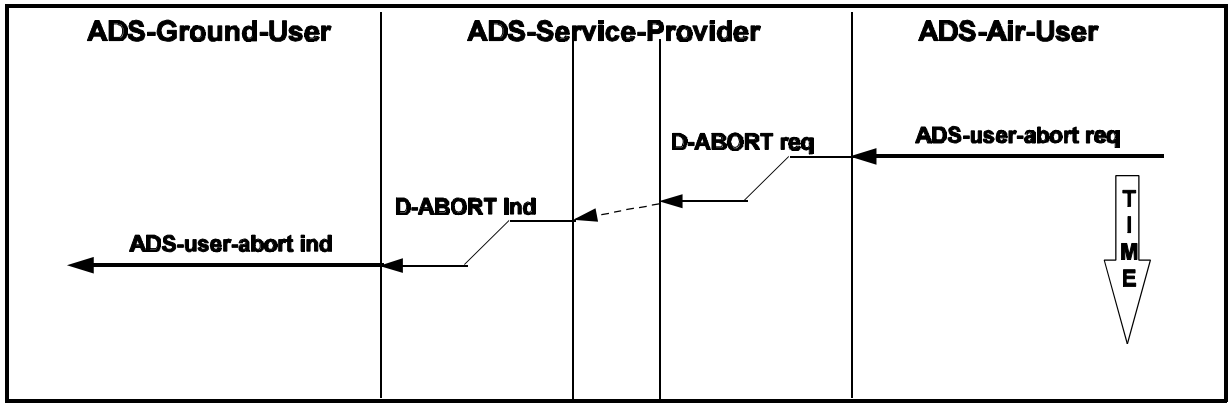

Figure 2.2.1.5-31: Air user abort service

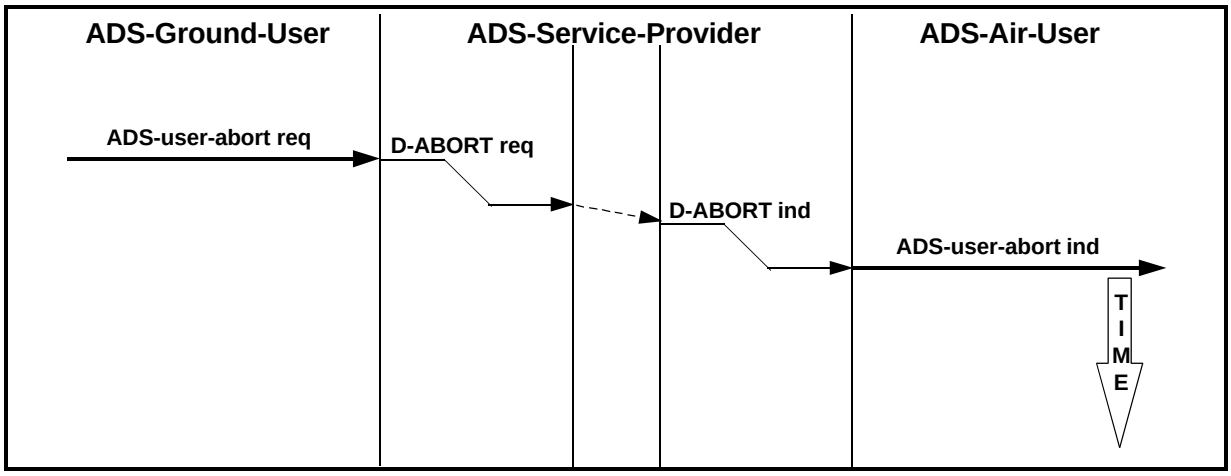

Figure 2.2.1.5-32: Ground user abort service

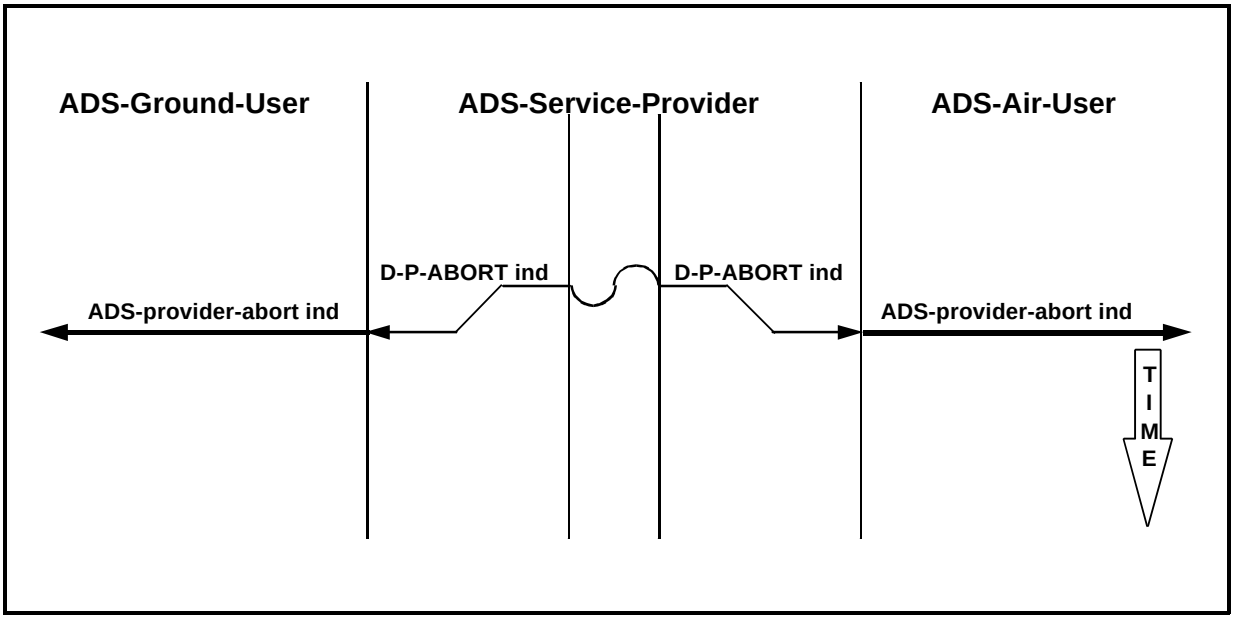

Figure 2.2.1.5-33: Dialogue service provider abort service

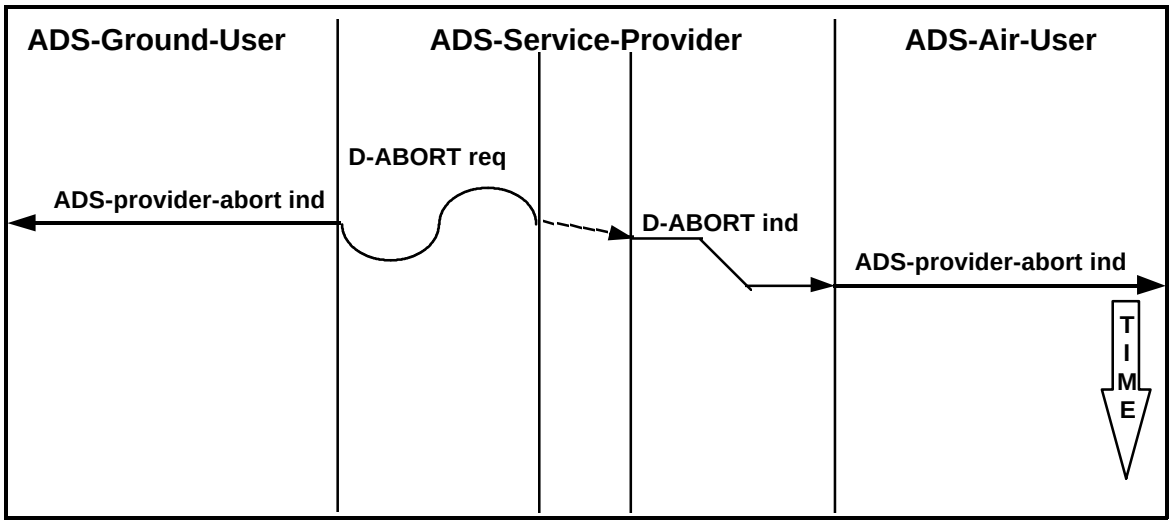

Figure 2.2.1.5-34: Ground ASE abort

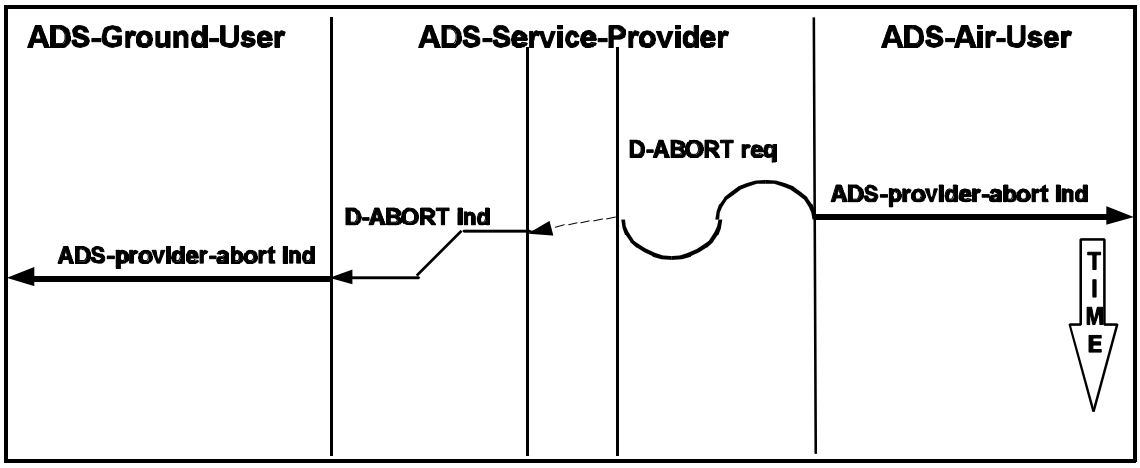

Figure 2.2.1.5-35: Air ASE abort

2.2.1.5.2 ADS Service Provider Timers

2.2.1.5.2.1 The ADS-ASE shall be capable of detecting when a timer expires.

Note 1.— Table 2.2.1.5-1 lists the time constraints related to the ADS application. Each time constraint requires a timer to be set in the ADS protocol machine.

Note 2.— If the timer expires before the final event has occurred, the ADS ASE takes the appropriate action as defined in 2.2.1.5.4.1.

2.2.1.5.2.2 **Recommendation.**—*The timer values should be as indicated in Table 2.2.1.5-1.*

Table 2.2.1.5-1: ADS Service Provider Timers

ADS Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
ADS-demand -contract	t-DC-1	6 minutes	ADS-demand-contract request	ADS-demand-contract confirmation or ADS-report indication
	t-DC-2	3 minutes 30 seconds	ADS-demand-contract confirmation	ADS-report indication
ADS-event-contract	t-EC-1	6 minutes	ADS-event-contract request	ADS-event-contract confirmation or ADS-report indication
	t-EC-2	6 minutes	ADS-cancel request	ADS-cancel-contract confirmation
ADS-periodic-contract	t-PC-1	6 minutes	ADS-periodic-contract request	ADS-periodic-contract confirmation or ADS-report indication
	t-PC-2	reporting rate + 3 minutes	ADS-report indication or ADS-periodic-contract confirmation	ADS-report indication
	t-PC-3	6 minutes	ADS-cancel request	ADS-cancel-contract confirmation

ADS Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
ADS emergency contract	t-EM-1	reporting rate + 3 minutes	ADS-emergency-report indication	ADS-emergency-report indication
	t-EM-2	6 minutes	ADS-modify-emergency-contract request	ADS-modify-emergency-contract request or ADS-emergency-report indication
	t-EM-3	6 minutes	ADS-cancel-emergency request	Arrival of ADS-cancel-emergency PDU
General	t-LI-1	6 minutes	D-END request	D-END confirmation

Note.— The receipt of ADS-user-abort request, D-ABORT indication or D-P-ABORT indication are also timer stop events.

2.2.1.5.3 ADS-ASE Protocol Description

2.2.1.5.3.1 Description

2.2.1.5.3.1.1 ADS-ASE Functional Model

Note 1.— The ADS-ground-ASE is functionally made of 7 modules as shown in figure 2.2.1.5-34 and the ADS-air-ASE is functionally made of a similar 7 modules as shown in figure 2.2.1.5-35:

a) the High Interface Module (HI module). This module interfaces with the ASE-user through the abstract service interface as defined in 2.2.1.3.

b) the ADS Demand Contract Module (DC module): the DC module manages all demand contracts with a single ground system.

c) the ADS Event Contract Module (EC module): the EC module manages event contracts with a single ground system.

d) the ADS Periodic Contract Module (PC module): the PC module manages periodic contracts with a single ground system.

e) the ADS Emergency Module (EM module): the EM module manages emergency contracts with a single ground system.

f) the ADS Abort Module (AB module): the AB module handles aborts in case of irrecoverable error.

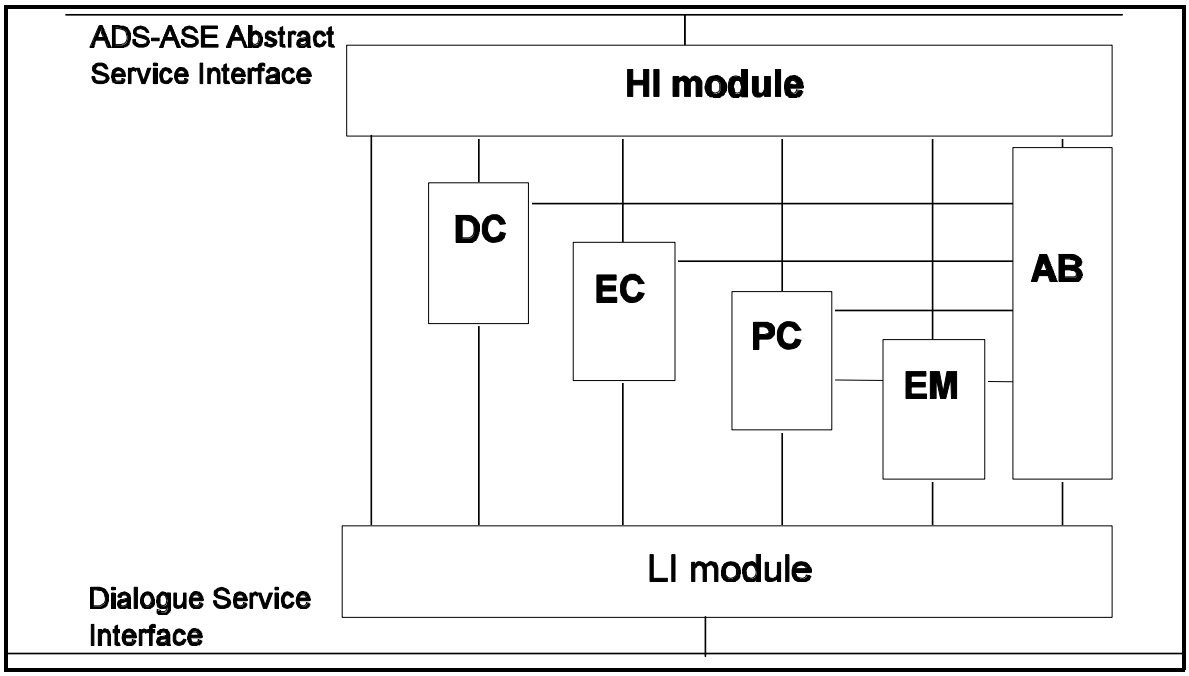

Figure 2.2.1.5-34: Functional model of the ADS-ground-ASE

g) the Low Interface Module (LI module). This module interfaces the Dialogue Service Provider on behalf of theDC, EC, PC, EM and AB modules.

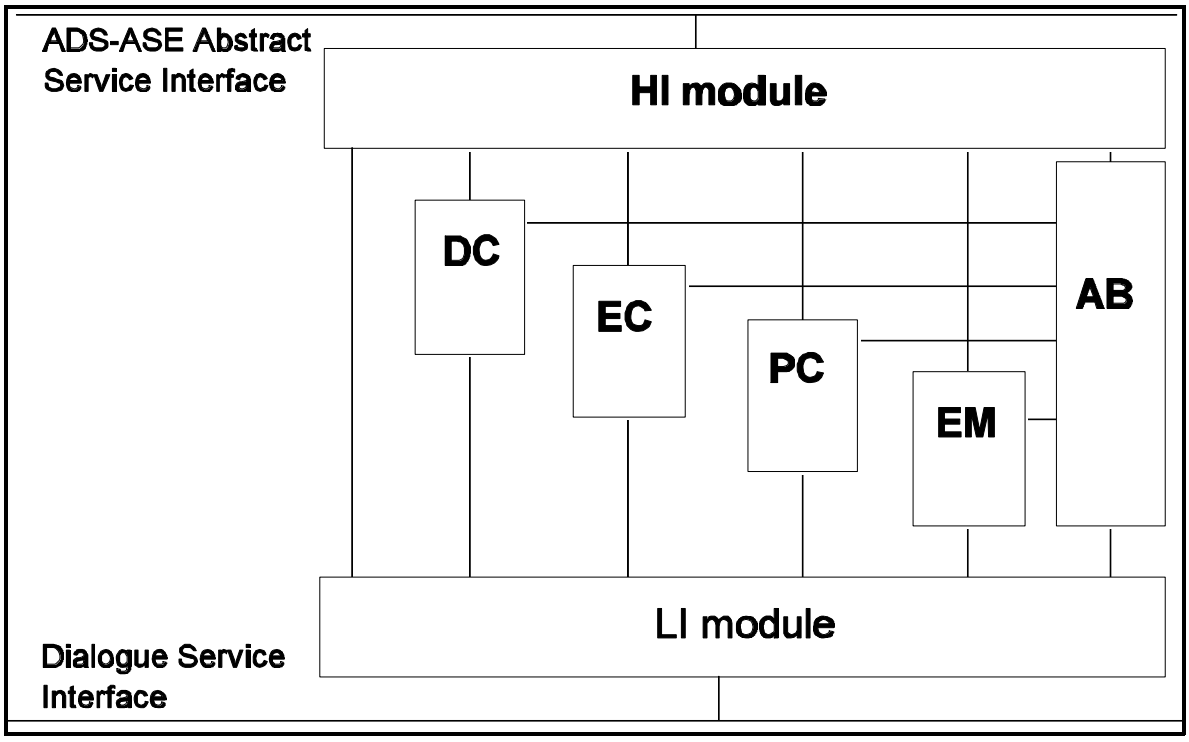

Figure 2.2.1.5-35: Functional model of the ADS-air-ASE

Note 2.— The only difference between the ADS-ground-ASE and the ADS-air-ASE functional models is that in the ADS-air-ASE, there is no communication between the PC and EM modules.

Note 3.— 2.2.1.5.3 describes the actions of the individual modules in both the air and ground systems. 2.2.1.5.6 contains state Tables for the individual modules.

Note 4.— The ADS-ground-user is considered an active user from the time at which it invokes the first ADS-demand-contract request, an ADS-event-contract request or an ADS-periodic-contract request until such time that:

a) the ADS-ground-user receives an ADS-cancel-all-contracts confirmation,

b) the ADS-ground-user receives an ADS-cancel confirmation, and there are no other contracts in place,

c) the ADS-ground-user receives an ADS-cancel-emergency-contract indication, and there are no other contracts in place,

d) the ADS-ground-user receives an ADS-demand-contract confirmation, an ADS-event-contract confirmation or an ADS-periodic-contract confirmation, with the Reply parameter value set to "negative acknowledgement", and there are no other contracts in place,

e) the ADS-ground-user receives an ADS-report indication, with the Contract type parameter value set to "demand contract", and there are no other contracts in place,

f) the ADS-ground-user receives an ADS-user-abort indication,

g) the ADS-ground-user receives an ADS-provider-abort indication, or

h) the ADS-ground-user invokes an ADS-user-abort request.

Note 5.— The ADS-air-user is considered an active user from the time at which it receives the first ADS-demand-contract indication, an ADS-event-contract indication or an ADS-periodic-contract indication until such time that:

a) the ADS-air-user receives an ADS-cancel-all-contracts indication,

b) the ADS-air-user receives an ADS-cancel indication, and there are no other contracts in place,

c) the ADS-air-user invokes an ADS-cancel-emergency-contract request, and there are no other contracts in place,

d) the ADS-air-user invokes an ADS-demand-contract response, an ADS-event-contract response or an ADS-periodic-contract response, with the Reply parameter value set to "negative acknowledgement", and there are no other contracts in place,

e) the ADS-air-user invokes an ADS-report request, with the Contract type parameter value set to "demand contract", and there are no other contracts in place,

f) the ADS-air-user receives an ADS-user-abort indication,

g) the ADS-air-user receives an ADS-provider-abort indication, or

h) the ADS-air-user invokes an ADS-user-abort request.

2.2.1.5.3.2 In 2.2.1.5.3, if no actions are described for an ADS service primitive in a particular state, then the invocation of that service primitive shall be prohibited in that state.

2.2.1.5.3.3 Possible errors arising upon Receipt of an APDU

2.2.1.5.3.3.1 If an APDU is not received when one is required, or one is received in an inappropriate dialogue service primitive, then exception handling procedures as described in 2.2.1.5.4.3 shall apply.

2.2.1.5.3.3.2 Upon receipt of an APDU, if no actions are described for the arrival of that APDU when in a particular state, then exception handling procedures as described in 2.2.1.5.4.4 shall apply.

2.2.1.5.3.3.3 Upon receipt of an APDU that cannot be decoded, then exception handling procedures as described in 2.2.1.5.4.7 shall apply.

2.2.1.5.3.4 Ground ADS HI Module

2.2.1.5.3.4.1 Upon receipt of a service primitive, the HI module shall pass it to the module as shown in Table 2.2.1.5-2.

Table 2.2.1.5-2: Request and response primitive to ground module mapping

Service Primitive	ADS-ground-ASE Module
ADS-demand-contract request	DC
ADS-event-contract request	EC
ADS-periodic-contract request	PC
ADS-cancel request with contract type event-contract	EC

ADS-cancel request with contract type periodic-contract	PC
ADS-cancel-all-contracts request	LI
ADS-modify-emergency-contract request	EM
ADS-user-abort request	AB

2.2.1.5.3.4.2 Upon receipt of a request to invoke a service primitive from one of the ground modules in the ADS-ground-ASE as shown in Table 2.2.1.5-3, the ground HI module shall do so.

Table 2.2.1.5-3: Indication and confirmation primitive to ground module mapping

ADS-ground-ASE Module	Service Primitive
DC	ADS-demand-contract confirmation
EC	ADS-event-contract confirmation
PC	ADS-periodic-contract confirmation
DC	ADS-report indication
EC	ADS-report indication
PC	ADS-report indication
EC	ADS-cancel confirmation
PC	ADS-cancel confirmation
LI	ADS-cancel-all-contracts confirmation
EM	ADS-emergency-report indication
EM	ADS-cancel-emergency indication
EM	ADS-modify-emergency-contract confirmation

2.2.1.5.3.4.3 On receipt of a request to invoke ADS-provider-abort indication from the ground AB module, the ground HI module shall:

a) if the ADS-ground-user is not an active user, take no further action;

b) if the ADS-ground-user is an active user, invoke ADS-provider-abort indication.

2.2.1.5.3.4.4 On receipt of a request to invoke ADS-user-abort indication from the ground AB module, the ground HI module shall:

a) if the ADS-ground-user is not an active user, take no further action;

b) if the ADS-ground-user is an active user, invoke ADS-user-abort indication.

2.2.1.5.3.4.5 The ground HI module shall reject requests and responses, apart from ADS-user-abort requests, when the ground LI module is in the LI-G-START state or the LI-G-END state.

2.2.1.5.3.5 Air ADS HI Module

2.2.1.5.3.5.1 Upon receipt of a service primitive, the air HI module shall pass it to the air module as shown in Table 2.2.1.5-4.

Table 2.2.1.5-4: Request and response primitive to air module mapping

Service Primitive	ADS-air-ASE Module
ADS-demand-contract response	DC
ADS-event-contract response	EC
ADS-periodic-contract response	PC
ADS-report request with *contract type* demand-contract	DC
ADS-report request with *contract type* event-contract	EC
ADS-report request with *contract type* periodic-contract	PC
ADS-emergency-report request	EM
ADS-cancel-emergency request	EM
ADS-modify-emergency-contract response	EM
ADS-user-abort request	AB

2.2.1.5.3.5.2 Upon receipt of a request to invoke a service primitive from one of the air modules in the ADS-air-ASE as shown in Table 2.2.1.5-5, the air HI module shall do so.

Table 2.2.1.5-5: Indication and confirmation primitive to air module mapping

ADS-air-ASE Module	Service Primitive
DC	ADS-demand-contract indication
EC	ADS-event-contract indication
PC	ADS-periodic-contract indication
EC	ADS-cancel indication
PC	ADS-cancel indication
LI	ADS-cancel-all-contracts indication

EM	ADS-modify-emergency-contract indication

2.2.1.5.3.5.3 On receipt of a request to invoke ADS-provider-abort indication from the air AB module, the air HI module shall:

a) if the ADS-air-user is not an active user, take no further action;

b) if the ADS-air-user is an active user, invoke ADS-provider-abort indication.

2.2.1.5.3.5.4 On receipt of a request to invoke ADS-user-abort indication from the air AB module, the air HI module shall:

a) if the ADS-air-user is not an active user, take no further action;

b) if the ADS-air-user is an active user, invoke ADS-user-abort indication.

2.2.1.5.3.6 Ground ADS DC Module

Note.— The states defined for the ground ADS DC module are the following:

a) DC-G-IDLE

b) DC-G-PENDING

c) DC-G-ACTIVE

2.2.1.5.3.6.1 On initiation, the ground DC module shall be in the DC-G-IDLE state.

2.2.1.5.3.6.2 Upon receipt of an ADS-demand-contract request:

2.2.1.5.3.6.2.1 If in the DC-G-IDLE state, the ground DC module shall:

a) create an ADS-demand-contract-PDU with elements derived as in Table 2.2.1.5-6,

b) pass it, together with the *aircraft identifier* parameter value, *ICAO facility designation* parameter value and the *class of communication service* parameter value, to the ground LI module,

c) start timer t-DC-1, and

d) enter the DC-G-PENDING state.

Table 2.2.1.5-6

PDU Element Name	Derivation of Element Value
ADS-demand-contract-PDU	*contract details* parameter

2.2.1.5.3.6.3 Upon receipt of an ADS-demand-report-PDU containing a *positive-acknowledgement* element:

2.2.1.5.3.6.3.1 If in the DC-G-PENDING state, the ground DC module shall:

a) stop the t-DC-1 timer,

b) request the ground HI module to invoke ADS-report indication with parameter values derived as in Table 2.2.1.5-7,

c) enter the DC-G-IDLE state.

Table 2.2.1.5-7

Parameter Name	Derivation of Parameter Value
Contract Type	"Demand contract"
Event Type	not supplied
Positive Acknowledgement	NULL
Report Details	*report* element of the ADS-demand-contract-PDU

2.2.1.5.3.6.4 Upon receipt of an ADS-demand-report-PDU not containing a *positive-acknowledgement* element:

2.2.1.5.3.6.4.1 If in the DC-G-ACTIVE state, the ground DC module shall:

a) stop the t-DC-2 timer,

b) request the ground HI module to invoke ADS-report indication with parameter values derived as in Table 2.2.15-8, and

c) enter the DC-G-IDLE state.

Table 2.2.1.5-8

Parameter Name	Derivation of Parameter Value
Contract Type	"Demand contract"
Event Type	not supplied

Parameter Name	Derivation of Parameter Value
Positive Acknowledgement	not supplied
Report Details	*report* element of the ADS-demand-contract-PDU

2.2.1.5.3.6.5 Upon receipt of an ADS-negative-acknowledgement-PDU:

2.2.1.5.3.6.5.1 If in the DC-G-PENDING state, the ground DC module shall:

a) stop the t-DC-1 timer,

b) request the ground HI module to invoke ADS-demand-contract confirmation with parameter values derived as in Table 2.2.1.5-9, and

c) enter the DC-G-IDLE state.

Table 2.2.1.5-9

Parameter Name	Derivation of Parameter Value
Reply	*NegativeAcknowledgement* with a value derived from the *reason* element of the ADS-negative-acknowledgement-PDU

2.2.1.5.3.6.6 Upon receipt of an ADS-noncompliance-notification-PDU:

2.2.1.5.3.6.6.1 If in the DC-G-PENDING state, the ground DC module shall:

a) stop the t-DC-1 timer,

b) request the ground HI module to invoke an ADS-demand-contract confirmation with parameter values derived as in Table 2.2.1.5-10,

c) start the t-DC-2 timer, and

d) enter the DC-G-ACTIVE state.

Table 2.2.1.5-10

Parameter Name	Derivation of Parameter Value
Reply	*NoncomplianceNotification* with a value derived from the *demand-ncn* element of the ADS-noncompliance-notification-PDU

2.2.1.5.3.6.6.2 Upon expiry of the t-DC-1 timer or the t-DC-2 timer, the ground DC module shall:

a) request the ground AB module to abort with reason *timer-expiry*, and

b) enter the DC-G-IDLE state

2.2.1.5.3.6.6.3 Upon receipt of a request from the ground AB or ground LI module to stop operation, the ground DC module shall:

a) stop any timers, and

b) enter the DC-G-IDLE state.

2.2.1.5.3.7 Air ADS DC Module

Note.— The states defined for the air ADS DC module are the following:

a) DC-A-IDLE

b) DC-A-PENDING

c) DC-A-ACTIVE

2.2.1.5.3.7.1 On initiation, the air DC module shall be in the DC-A-IDLE state.

2.2.1.5.3.7.2 Upon receipt of an ADS-demand-contract response with the *Reply* parameter value set to *negative acknowledgement*:

2.2.1.5.3.7.2.1 If in the DC-A-PENDING state, the air DC module shall:

a) create an ADS-negative-acknowledgement-PDU with elements as defined in Table 2.2.1.5-11,

b) pass it to the air LI module, and

c) enter the DC-A-IDLE state.

Table 2.2.1.5-11

PDU Element Name	Derivation of Element Value
request-type	demand-contract
reason	*Reply* parameter value

2.2.1.5.3.7.3 Upon receipt of an ADS-demand-contract response with the *reply* parameter value set to *noncompliance notification*:

2.2.1.5.3.7.3.1 If in the DC-A-PENDING state, the air DC module shall:

a) create an ADS-noncompliance-notification-PDU with elements as defined in Table 2.2.1.5-12,

b) pass it to the air LI module, and

c) enter the DC-A-ACTIVE state.

Table 2.2.1.5-12

PDU Element Name	Derivation of Element Value
NoncomplianceNotification	demand-ncn with a value derived from *Reply* parameter value

2.2.1.5.3.7.4 Upon receipt of an ADS-report request:

2.2.1.5.3.7.4.1 If in the DC-A-PENDING state and the *positive acknowledgement* parameter is present, the air DC module shall:

a) create ADS-demand-report-PDU with a value derived as in Table 2.2.1.5-13,

b) pass it to the air LI module, and

c) enter the DC-A-IDLE state.

2.2.1.5.3.7.4.2 If in the DC-A-ACTIVE state and the *positive acknowledgement* parameter is not present, the air DC module shall:

a) create ADS-demand-report-PDU with a value derived as in Table 2.2.1.5-14,

b) pass it to the air LI module, and

c) enter the DC-A-IDLE state.

Table 2.2.1.5-13

PDU Element Name	Derivation of Element Value
Report	*Report details* parameter value
Positive acknowledgement	NULL

Table 2.2.1.5-14

PDU Element Name	Derivation of Element Value
Report	*Report details* parameter value
Positive acknowledgement	not supplied

2.2.1.5.3.7.5 Upon receipt of an ADS-demand-contract-PDU:

2.2.1.5.3.7.5.1 If in the DC-A-IDLE state, the air DC module shall:

a) request the air HI module to invoke ADS-demand-contract indication with parameters derived as in Table 2.2.1.5-15, and,

b) enter the DC-A-PENDING state.

Table 2.2.1.5-15

PDU Element Name	Derivation of Element Value
Contract details	ADS-demand-contract-PDU
ICAO facility designation	Calling peer id, if provided by the air LI module

2.2.1.5.3.7.5.2 Upon receipt of a request from the air AB or air LI module to stop operation, the air DC module shall enter the DC-A-IDLE state.

2.2.1.5.3.8 Ground ADS EC Module

Note.— The states defined for the ground ADS EC module are the following:

a) EC-G-IDLE

b) *EC-G-START-PENDING*

c) *EC-G-ACTIVE*

d) *EC-G-PENDING*

e) *EC-G-CANCEL*

2.2.1.5.3.8.1 On initiation, the ground EC module shall be in the EC-G-IDLE state.

2.2.1.5.3.8.2 Upon receipt of an ADS-event-contract request:

2.2.1.5.3.8.2.1 If in the EC-G-IDLE state, the ground EC module shall:

a) create an ADS-event-contract-PDU with elements as defined in Table 2.2.1.5-16,

b) pass it, together with the *aircraft identifier* parameter value, *ICAO facility designation* parameter value and the *class of communication service* parameter value to the ground LI module,

c) start timer t-EC-1, and

d) enter the EC-G-START-PENDING state.

2.2.1.5.3.8.2.2 If in the EC-G-ACTIVE state, the ground EC module shall:

a) create an ADS-event-contract-PDU with elements as defined in Table 2.2.1.5-17,

b) pass it to the ground LI module,

c) start timer t-EC-1, and

d) enter the EC-G-PENDING state.

Table 2.2.1.5-16

PDU Element Name	Derivation of Element Value
ADS-event-contract-PDU	*contract details* parameter value

Table 2.2.1.5-17

PDU Element Name	Derivation of Element Value
ADS-event-contract-PDU	*contract details* parameter value

2.2.1.5.3.8.3 Upon receipt of an ADS-cancel request:

2.2.1.5.3.8.3.1 If in the EC-G-ACTIVE state, the ground EC module shall:

a) create an ADS-cancel-contract-PDU with elements as defined in Table 2.2.1.5-18,

b) pass it to the ground LI module,

c) start timer t-EC-2, and

d) enter the EC-G-CANCEL state.

Table 2.2.1.5-18

PDU Element Name	Derivation of Element Value
CancelContract	event-contract

2.2.1.5.3.8.4 Upon receipt of an ADS-event-report-PDU containing a *positive acknowledgement*:

2.2.1.5.3.8.4.1 If in the EC-G-PENDING or the EC-G-START-PENDING state, the ground EC module shall:

a) stop the t-EC-1 timer,

b) request the ground HI module to invoke ADS-report indication, with parameter values derived as in Table 2.2.1.5-19, and

c) enter the EC-G-ACTIVE state.

Table 2.2.1.5-19

Parameter Name	Derivation of Parameter Value
Contract type	“event contract”
Event type	event-type PDU element
Positive acknowledgement	NULL
Report details	report PDU element

2.2.1.5.3.8.5 Upon receipt of an ADS-event-report-PDU not containing a *positive acknowledgement*:

2.2.1.5.3.8.5.1 If in the EC-G-ACTIVE, EC-G-PENDING or EC-G-CANCEL state, the ground EC module shall:

a) request the ground HI module to invoke ADS-report indication, with parameter values derived as in Table 2.2.1.5-20, and

b) remain in the same state.

Table 2.2.1.5-20

Parameter Name	Derivation of Parameter Value
Contract type	"event contract"
Event type	event-type PDU element
Positive acknowledgement	not supplied
Report details	report PDU element

2.2.1.5.3.8.6 Upon receipt of an ADS-positive-acknowledgement-PDU for an event contract:

2.2.1.5.3.8.6.1 If in the EC-G-START-PENDING state or the EC-G-PENDING state, the ground EC module shall:

a) stop the t-EC-1 timer,

b) request the ground HI module to invoke ADS-event-contract confirmation, with parameter values derived as in Table 2.2.1.5-21, and

c) enter the EC-G-ACTIVE state.

Table 2.2.1.5-21

Parameter Name	Derivation of Parameter Value
Reply	*PositiveAcknowledgement* with abstract value NULL

2.2.1.5.3.8.7 Upon receipt of an ADS-positive-acknowledgement-PDU for a cancel contract:

2.2.1.5.3.8.7.1 If in the EC-G-CANCEL state, the ground EC module shall:

a) stop the t-EC-2 timer,

b) request the ground HI module to invoke ADS-cancel-contract confirmation, with parameter values derived as in Table 2.2.1.5-22, and

c) enter the EC-G-IDLE state.

Table 2.2.1.5-22

Parameter Name	Derivation of Parameter Value
Contract type	event-contract

2.2.1.5.3.8.8 Upon receipt of an ADS-negative-acknowledgement-PDU:

2.2.1.5.3.8.8.1 If in the EC-G-START-PENDING state, the ground EC module shall:

a) stop the t-EC-1 timer,

b) request the ground HI module to invoke ADS-event-contract confirmation, with parameter values derived as in Table 2.2.1.5-23, and

c) enter the EC-G-IDLE state.

2.2.1.5.3.8.8.2 If in the EC-G-PENDING state, the ground EC module shall:

a) stop the t-EC-1 timer,

b) request the ground HI module to invoke ADS-event-contract confirmation, with parameter values derived as in Table 2.2.1.5-23, and

c) enter the EC-G-ACTIVE state.

Table 2.2.1.5-23

Parameter Name	Derivation of Parameter Value
Reply	*NegativeAcknowledgement* with value derived from reason PDU element

2.2.1.5.3.8.9 Upon receipt of an ADS-noncompliance-notification-PDU for an event contract:

2.2.1.5.3.8.9.1 If in the EC-G-START-PENDING state or the EC-G-PENDING, the ground EC module shall:

a) stop the t-EC-1 timer,

b) request the ground HI module to invoke ADS-event-contract confirmation, with parameter values derived as in Table 2.2.1.5-24, and

c) enter the EC-G-ACTIVE state.

Table 2.2.1.5-24

Parameter Name	Derivation of Parameter Value
Reply	*NoncomplianceNotification* with value derived from event-ncn PDU element

2.2.1.5.3.8.9.2 Upon expiry of the t-EC-1 timer or the t-EC-2 timer, the ground EC module shall:

a) request the ground AB module to abort with reason *timer-expiry*, and

b) enter the EC-G-IDLE state

2.2.1.5.3.8.9.3 Upon receipt of a request from the ground AB module to stop operation, the EC module shall:

a) stop any timers, and

b) enter the EC-G-IDLE state.

2.2.1.5.3.9 Air ADS EC Module

Note.— The states defined for the air ADS EC module are the following:

a) EC-A-IDLE

b) EC-A-PENDING

c) EC-A-ACTIVE

d) EC-A-ACTIVE-PENDING

2.2.1.5.3.9.1 On initiation, the air EC module shall be in the EC-A-IDLE state.

2.2.1.5.3.9.2 Upon receipt of an ADS-event-contract response with the *reply* parameter value set to *positive acknowledgement*:

2.2.1.5.3.9.2.1 If in the EC-A-PENDING state or in the EC-A-ACTIVE-PENDING state, the air EC module shall:

a) create an ADS-positive-acknowledgement-PDU with elements as defined in Table 2.2.1.5-25

b) pass it to the air LI module, and

c) enter the EC-A-ACTIVE state.

Table 2.2.1.5-25

PDU Element Name	Derivation of Element Value
Request type	event-contract

2.2.1.5.3.9.3 Upon receipt of an ADS-event-contract response with the *reply* parameter value set to *noncompliance notification*:

2.2.1.5.3.9.3.1 If in the EC-A-PENDING state or in the EC-A-ACTIVE-PENDING state, the air EC module shall:

a) create an ADS-noncompliance-notification-PDU with elements as defined in Table 2.2.1.5-26,

b) pass it to the air LI module, and

c) enter the EC-A-ACTIVE state.

Table 2.2.1.5-26

PDU Element Name	Derivation of Element Value
NoncomplianceNotification	*NoncomplianceNotification* with value derived from *Reply* parameter

2.2.1.5.3.9.4 Upon receipt of an ADS-event-contract response with the *reply* parameter value set to *negative acknowledgement*:

2.2.1.5.3.9.4.1 If in the EC-A-PENDING state, the air EC module shall:

a) create an ADS-negative-acknowledgement-PDU with elements as defined in Table 2.2.1.5-27,

b) pass it to the air LI module, and

c) enter the EC-A-IDLE state.

2.2.1.5.3.9.4.2 If in the EC-A-ACTIVE-PENDING state, the air EC module shall:

a) create an ADS-negative-acknowledgement-PDU with elements as defined in Table 2.2.1.5-27,

b) pass it to the air LI module, and

c) enter the EC-A-ACTIVE state.

Table 2.2.1.5-27

PDU Element Name	Derivation of Element Value
Request Type	event-contract
Reason	*Reply* parameter value

2.2.1.5.3.9.5 Upon receipt of an ADS-report request with a positive acknowledgement parameter:

2.2.1.5.3.9.5.1 If in the EC-A-PENDING state or the EC-A-ACTIVE-PENDING state, the air EC module shall:

a) create an ADS-event-report-PDU with element as defined in Table 2.2.1.5-28,

b) pass it to the air LI module, and

c) enter the EC-A-ACTIVE state.

Table 2.2.1.5-28

PDU Element Name	Derivation of Element Value
Event type	*Event type* parameter value
Report	*Report details* parameter value
Positive Acknowledgement	NULL

2.2.1.5.3.9.6 Upon receipt of an ADS-report request with no *positive acknowledgement* parameter:

2.2.1.5.3.9.6.1 If in the EC-A-ACTIVE state, the air EC module shall:

a) create an ADS-event-report-PDU with element as defined in Table 2.2.1.5-29,

b) pass it to the air LI module, and

c) remain in the EC-A-ACTIVE state.

Table 2.2.1.5-29

PDU Element Name	Derivation of Element Value
Event type	*Event type* parameter value
Report	*Report details* parameter value
Positive Acknowledgement	not supplied

2.2.1.5.3.9.7 Upon receipt of an ADS-event-contract-PDU:

2.2.1.5.3.9.7.1 If in the EC-A-IDLE state, the air EC module shall:

a) request the air HI module to invoke ADS-event-contract indication with parameter values derived as in Table 2.2.1.5-30, and

b) enter the EC-A-PENDING state.

2.2.1.5.3.9.7.2 If in the EC-A-ACTIVE state, the air EC module shall:

a) request the air HI module to invoke ADS-event-contract indication with parameter values derived as in Table 2.2.1.5-30, and

b) enter the EC-A-ACTIVE-PENDING state.

Table 2.2.1.5-30

Parameter Name	Derivation of Parameter Value
Contract details	ADS-event-contract-PDU
ICAO facility designation	Calling peer id, if provided by the air LI module

2.2.1.5.3.9.8 Upon receipt of an ADS-cancel-contract-PDU:

2.2.1.5.3.9.8.1 If in the EC-A-ACTIVE state, the air EC module shall:

a) request the HI module to invoke ADS-cancel indication (event-contract) with parameter values as defined in Table 2.2.1.5-31,

b) create an ADS-positive-acknowledgement-PDU (cancel-event-contract) with elements as defined in Table 2.2.1.5-32,

c) pass it to the air LI module, and

d) enter the EC-A-IDLE state.

Table 2.2.1.5-31

Parameter Name	Derivation of Parameter Value
Contract type	cancel-event-contract

Table 2.2.1.5-32

PDU Element Name	Derivation of Element Value
Request type	event-contract

2.2.1.5.3.9.8.2 Upon receipt of a request from the air AB or air LI module to stop operation, the air EC module shall enter the EC-A-IDLE state.

2.2.1.5.3.10 Ground ADS PC Module

Note.— The states defined for the ground ADS PC module are the following:

a) PC-G-IDLE

b) PC-G-START-PENDING

c) PC-G-ACTIVE

d) PC-G-PENDING

e) PC-G-CANCEL

2.2.1.5.3.10.1 On initiation, the ground PC module shall be in the PC-G-IDLE state.

Note.— The ground PC module has a boolean variable named EMERGENCY.

2.2.1.5.3.10.2 On initiation, EMERGENCY shall be set to FALSE.

2.2.1.5.3.10.3 Upon receipt of an ADS-periodic-contract request:

2.2.1.5.3.10.3.1 If in the PC-G-IDLE state, the ground PC module shall:

a) create an ADS-periodic-contract-PDU with elements as defined in Table 2.2.1.5-33,

b) pass it, together with the *aircraft identifier* parameter value, *ICAO facility designation* parameter value and the *Class of communication service* parameter value, to the ground LI module,

c) start timer t-PC-1, and

d) enter the PC-G-START-PENDING state.

2.2.1.5.3.10.3.2 If in the PC-G-ACTIVE state, the PC module shall:

a) if EMERGENCY = FALSE, stop the t-PC-2 timer,

b) create an ADS-periodic-contract-PDU with elements as defined in Table 2.2.1.5-33,

c) pass it, the aircraft identifier parameter value, and the *Class of communication service* parameter value, to the ground LI module,

d) start timer t-PC-1, and

e) enter the PC-G-PENDING state.

Table 2.2.1.5-33

PDU Element Name	Derivation of Element Value
ADS-periodic-contract	*Contract details* parameter value

2.2.1.5.3.10.4 Upon receipt of an ADS-cancel request:

2.2.1.5.3.10.4.1 If in the PC-G-ACTIVE state, the ground PC module shall:

a) stop the t-PC-2 timer,

b) create an ADS-cancel-PDU with elements as defined in Table 2.2.1.5-34,

c) pass it to the ground LI module,

d) start timer t-PC-3, and

e) enter the PC-G-CANCEL state.

Table 2.2.1.5-34

PDU Element Name	Derivation of Element Value
ADS-cancel-contract-PDU	periodic-contract

2.2.1.5.3.10.5 Upon receipt of an ADS-periodic-report-PDU containing a *positive acknowledgement*:

2.2.1.5.3.10.5.1 If in the PC-G-START-PENDING state, or the PC-G-PENDING state, the ground PC module shall:

a) stop the t-PC-1 timer,

b) create the *report details* parameter of an ADS-report indication,

c) request the ground HI module to invoke ADS-report indication with parameter values derived as in Table 2.2.1.5-35,

d) if EMERGENCY = FALSE, start the t-PC-2 timer, and

e) enter the PC-G-ACTIVE state.

Table 2.2.1.5-35

Parameter Name	Derivation of Parameter Value
Contract type	periodic-contract
Event type	not supplied
Positive acknowledgement	NULL
Report details	report PDU element

2.2.1.5.3.10.6 Upon receipt of an ADS-periodic-report-PDU not containing a *positive acknowledgement*:

2.2.1.5.3.10.6.1 If in the PC-G-PENDING state, the ground PC module shall:

a) request the ground HI module to invoke ADS-report indication with parameter values derived as in Table 2.2.1.5-36,

b) remain in the PC-G-PENDING state.

2.2.1.5.3.10.6.2 If in the PC-G-ACTIVE state, the ground PC module shall:

a) if EMERGENCY = FALSE, stop the t-PC-2 timer,

b) request the ground HI module to invoke ADS-report indication with parameter values derived as in Table 2.2.1.5-36,

c) if EMERGENCY = FALSE, start the t-PC-2 timer, and

d) remain in the PC-G-ACTIVE state.

2.2.1.5.3.10.6.3 If in the PC-G-CANCEL state, the ground PC module shall:

a) request the ground HI module to invoke ADS-report indication with parameter values derived as in Table 2.2.1.5-36, and

b) remain in the PC-G-CANCEL state.

Table 2.2.1.5-36

Parameter Name	Derivation of Parameter Value
Contract type	periodic-contract
Event type	not supplied
Positive acknowledgement	not supplied
Report details	report PDU element

2.2.1.5.3.10.7 Upon receipt of an ADS-positive-acknowledgement-PDU for a periodic contract:

2.2.1.5.3.10.7.1 If in the PC-G-START-PENDING state, or the PC-G-PENDING state, the ground PC module shall:

a) stop the t-PC-1 timer,

b) request the ground HI module to invoke ADS-periodic-contract confirmation with parameter values derived as in Table 2.2.1.5-37,

c) if EMERGENCY = FALSE, start the t-PC-2 timer, and

d) enter the PC-G-ACTIVE state.

Table 2.2.1.5-37

Parameter Name	Derivation of Parameter Value
Reply	*PositiveAcknowledgement* with value NULL

2.2.1.5.3.10.8 Upon receipt of an ADS-positive-acknowledgement-PDU for an cancel contract:

2.2.1.5.3.10.8.1 If in the PC-G-CANCEL state, the ground PC module shall:

a) stop the t-PC-3 timer,

b) request the ground HI module to invoke ADS-cancel-contract confirmation with parameter values derived as in Table 2.2.1.5-38, and

c) enter the PC-G-IDLE state.

Table 2.2.1.5-38

Parameter Name	Derivation of Parameter Value
Contract type	periodic-contract

2.2.1.5.3.10.9 Upon receipt of an ADS-negative-acknowledgement-PDU:

2.2.1.5.3.10.9.1 If in the PC-G-START-PENDING state, the ground PC module shall:

a) stop the t-PC-1 timer,

b) request the ground HI module to invoke ADS-periodic-contract confirmation with parameter values derived as in Table 2.2.1.5-39, and

c) enter the PC-G-IDLE state.

2.2.1.5.3.10.9.2 If in the PC-G-PENDING state, the ground PC module shall:

a) stop the t-PC-1 timer,

b) request the ground HI module to invoke ADS-periodic-contract confirmation with parameter values derived as in Table 2.2.1.5-39, and

c) if EMERGENCY = FALSE, start the t-PC-2 timer, and

d) enter the PC-G-ACTIVE state.

Table 2.2.1.5-39

Parameter Name	Derivation of Parameter Value
Reply	*NegativeAcknowledgement* with value derived from the reason PDU element

2.2.1.5.3.10.10 Upon receipt of an ADS-noncompliance-notification-PDU for a periodic contract:

2.2.1.5.3.10.10.1 If in the PC-G-START-PENDING state, or the PC-G-PENDING state, the ground PC module shall:

a) stop the t-PC-1 timer,

b) request the ground HI module to invoke ADS-periodic-contract confirmation with parameter values derived as in Table 2.2.15-40, and

c) if EMERGENCY = FALSE, start the t-PC-2 timer, and

d) enter the PC-G-ACTIVE state.

Table 2.2.1.5-40

Parameter Name	Derivation of Parameter Value
Reply	*NoncomplianceNotification* with value derived from the event-ncn PDU element

2.2.1.5.3.10.11 Upon receipt of a request to suspend periodic contracts from the ground EM module, the PC module shall:

a) if in the PC-G-ACTIVE state, stop the t-PC-2 timer,

b) set EMERGENCY to be TRUE, and

c) remain in the same state.

2.2.1.5.3.10.12 Upon receipt of a request to reinstate periodic contracts from the ground EM module, the ground PC module shall:

a) if in the PC-G-ACTIVE state, start the t-PC-2 timer, based on the most recent value of the period of the contract,

b) set EMERGENCY to be FALSE, and

c) remain in the same state.

2.2.1.5.3.10.13 Upon receipt of a request from the ground AB or ground LI module to stop operation, the ground PC module shall:

a) stop any timers,

b) set EMERGENCY to be FALSE, and

c) enter the PC-G-IDLE state.

2.2.1.5.3.10.14 Upon expiry of the t-PC-1 timer, t-PC-2 timer, or t-PC-3 timer, the ground PC module shall:

a) request the ground AB module to abort with reason timer-expiry, and

b) enter the PC-G-IDLE state.

2.2.1.5.3.11 Air ADS PC Module

Note.— The states defined for the air ADS PC module are the following:

a) PC-A-IDLE

b) PC-A-PENDING

c) PC-A-ACTIVE

d) PC-A-ACTIVE-PENDING

2.2.1.5.3.11.1 On initiation, the air PC module shall be in the PC-A-IDLE state.

2.2.1.5.3.11.2 Upon receipt of an ADS-periodic-contract response with the *reply* parameter value set to *positive acknowledgement* or *noncompliance notification* , then:

2.2.1.5.3.11.2.1 If in the PC-A-PENDING state or PC-A-ACTIVE-PENDING state, the air PC module shall:

a) If the *reply* parameter value is a *positive acknowledgement*, create an ADS-positive-acknowledgement-PDU with elements as defined in Table 2.2.1.5-41, or

b) If the *reply* parameter value is a *noncompliance notification*, create an ADS-noncompliance-notification-PDU with elements as defined in Table 2.2.1.5-42,

c) pass it to the air LI module, and

d) enter the PC-A-ACTIVE state.

Table 2.2.1.5-41

PDU Element Name	Derivation of Element Value
ADS-positive-acknowledgement-PDU	periodic-contract

Table 2.2.1.5-42

PDU Element Name	Derivation of Element Value
NoncomplianceNotification	*Reply* parameter value

2.2.1.5.3.11.3 Upon receipt of an ADS-periodic-contract response with the *reply* parameter value set to *negative acknowledgement*, then:

2.2.1.5.3.11.3.1 If in the PC-A-PENDING state, the air PC module shall:

a) create an ADS-negative-acknowledgement-PDU with elements as defined in Table 2.2.1.5-43,

b) pass it to the air LI module, and

c) enter the PC-A-IDLE state.

2.2.1.5.3.11.3.2 If in the PC-A-ACTIVE-PENDING state, the air PC module shall:

a) create an ADS-negative-acknowledgement-PDU with elements as defined in Table 2.2.1.5-43

b) pass it to the air LI module, and

c) enter the PC-A-ACTIVE state.

Table 2.2.1.5-43

PDU Element Name	Derivation of Element Value
request-type	periodic-contract
reason	*Reply* parameter value

2.2.1.5.3.11.4 Upon receipt of an ADS-report request with a *positive acknowledgement* parameter, then:

2.2.1.5.3.11.4.1 If in the PC-A-PENDING state or PC-A-ACTIVE-PENDING, the air PC module shall:

a) create ADS-periodic-report-PDU with elements as defined in Table 2.2.1.5-44,

b) pass it to the air LI module, and

c) enter the PC-A-ACTIVE state.

Table 2.2.1.5-44

PDU Element Name	Derivation of Element Value
report	*Report details* parameter value
positive-acknowledgement	NULL

2.2.1.5.3.11.5 Upon receipt of an ADS-report request with no *positive acknowledgement* parameter, then:

2.2.1.5.3.11.5.1 If in the PC-A-ACTIVE state, the air PC module shall:

a) create ADS-periodic-report-PDU with elements as defined in Table 2.2.1.5-45,

b) pass it to the air LI module, and

c) remain in the PC-A-ACTIVE state.

Table 2.2.1.5-45

PDU Element Name	Derivation of Element Value
report	*Report details* parameter value
positive-acknowledgement	not present

2.2.1.5.3.11.6 Upon receipt of an ADS-periodic-contract-PDU:

2.2.1.5.3.11.6.1 If in the PC-A-IDLE state, the air PC module shall:

a) request the air HI module to invoke ADS-periodic-contract indication with parameter values derived as in Table 2.2.1.5-46, and

b) enter the PC-A-PENDING state.

2.2.1.5.3.11.6.2 If in the PC-A-ACTIVE state, the air PC module shall:

a) request the air HI module to invoke ADS-periodic-contract indication with parameter values derived as in Table 2.2.1.5-46, and

b) enter the PC-A-ACTIVE-PENDING state.

Table 2.2.1.5-46

Parameter Name	Derivation of Parameter Value
Contract details	ADS-periodic-contract-PDU
ICAO facility designation	Calling peer id, if provided by the air LI module

2.2.1.5.3.11.7 Upon receipt of an ADS-cancel-contract-PDU, then:

2.2.1.5.3.11.7.1 If in the PC-A-ACTIVE state, the air PC module shall:

a) request the air HI module to invoke ADS-cancel indication with parameter values derived as in Table 2.2.1.5-47,

b) create an ADS-positive-acknowledgement-PDU with elements as defined in Table 2.2.1.5-48,

c) pass it to the air LI module, and

d) enter the PC-A-IDLE state.

Table 2.2.1.5-47

Parameter Name	Derivation of Parameter Value
Contract type	periodic-contract

Table 2.2.1.5-48

PDU Element Name	Derivation of Element Value
RequestType	cancel-periodic-contract

2.2.1.5.3.11.7.2 Upon receipt of a request from the air AB or air LI module to stop operation, the air PC module shall enter the PC-A-IDLE state.

2.2.1.5.3.12 Ground ADS EM Module

Note.— The states defined for the ground ADS EM module are the following:

a) *EM-G-IDLE*

b) *EM-G-ACTIVE*

c) *EM-G-MODIFY*

2.2.1.5.3.12.1 On initiation, the ground EM module shall be in the EM-G-IDLE state.

2.2.1.5.3.12.2 Upon receipt of an ADS-modify-emergency-contract request:

2.2.1.5.3.12.2.1 If in the EM-G-ACTIVE state, the ground EM module shall:

a) stop the t-EM-1 timer,

b) create an ADS-modify-emergency-contract-PDU with elements as definied in Table 2.2.1.5-49,

c) pass it to the ground LI module,

d) start the t-EM-2 timer, and

e) enter the EM-G-MODIFY state.

Table 2.2.1.5-49

PDU Element Name	Derivation of Element Value
ModifyEmergency	*reporting interval* parameter value

2.2.1.5.3.12.3 Upon receipt of an ADS-emergency-report-PDU containing a positive acknowledgement:

2.2.1.5.3.12.3.1 If in the EM-G-MODIFY state, the ground EM module shall:

a) stop the t-EM-2 timer,

b) request the ground HI module to invoke invoke ADS-emergency-report indication with parameter values derived as in Table 2.2.1.5-50,

c) start the t-EM-1 timer, and

d) enter the EM-G-ACTIVE state.

Table 2.2.1.5-50

Parameter Name	Derivation of Parameter Value

Positive acknowledgement of modification	NULL
Emergency report details	emergency-report element of ADSEmergency

2.2.1.5.3.12.4 Upon receipt of an ADS-emergency-report-PDU not containing a positive acknowledgement:

2.2.1.5.3.12.4.1 If in the EM-G-IDLE state, the ground EM module shall:

a) request the PC module to suspend operation,

b) request the ground HI module to invoke ADS-emergency-report indication with parameter values derived as in Table 2.2.1.5-51, and

c) start the t-EM-1 timer, and

d) enter the EM-G-ACTIVE state.

2.2.1.5.3.12.4.2 If in the EM-G-ACTIVE state, the ground EM module shall:

a) stop the t-EM-1 timer,

b) request the ground HI module to invoke ADS-emergency-report indication with parameter values derived as in Table 2.2.1.5-51, and

c) start the t-EM-1 timer, and

d) enter the EM-G-ACTIVE state.

2.2.1.5.3.12.4.3 If in the EM-G-MODIFY state, the ground EM module shall:

a) request the ground HI module to invoke ADS-emergency-report indication with parameter values derived as in Table 2.2.1.5-51, and

b) remain in the EM-G-MODIFY state.

Table 2.2.1.5-51

Parameter Name	Derivation of Parameter Value
Positive acknowledgement of modification	not provided
Emergency report details	emergency-report element of ADSEmergency

2.2.1.5.3.12.5 Upon receipt of an ADS-cancel-emergency-PDU:

2.2.1.5.3.12.5.1 If in the EM-G-ACTIVE state, the ground EM module shall:

a) stop the t-EM-1 timer,

b) request the ground HI module to invoke ADS-cancel-emergency indication,

c) create ADS-cancel-emergency-acknowledgement-PDU,

d) pass it to the ground LI module,

e) request the ground PC module to reinstate any periodic contracts, and

f) enter the EM-G-IDLE state.

2.2.1.5.3.12.5.2 If in the EM-G-MODIFY state, the ground EM module shall:

a) stop the t-EM-2 timer,

b) request the ground HI module to invoke ADS-cancel-emergency indication,

c) create ADS-cancel-emergency-acknowledgement-PDU,

d) pass it to the ground LI module,

e) request the ground PC module to reinstate any periodic contracts, and

f) enter the EM-G-IDLE state.

2.2.1.5.3.12.6 Upon receipt of an ADS-negative-acknowledgement-PDU for a modify-emergency-contract:

2.2.1.5.3.12.6.1 If in the EM-G-MODIFY state, the ground EM module shall:

a) stop the t-EM-2 timer,

b) request the ground HI module to invoke ADS-modify-emergency-contract confirmation,

c) start the t-EM-1 timer, and

d) enter the EM-G-ACTIVE state.

2.2.1.5.3.12.7 Upon expiry of the t-EM-1 timer or the t-EM-2 timer, the ground EM module shall:

a) request the ground AB module to abort with reason *timer-expiry*, and

b) enter the EM-G-IDLE state.

2.2.1.5.3.12.7.1 Upon receipt of a request from the ground AB or ground LI module to stop operation, the ground EM module shall:

a) stop any timers, and

b) enter the EM-G-IDLE state.

2.2.1.5.3.13 Air ADS EM Module

Note.— The states defined for the air ADS EM module are the following:

a) EM-A-IDLE

b) EM-A-ACTIVE

c) EM-A-MODIFY

d) EM-A-CANCEL

2.2.1.5.3.13.1 On initiation, the air EM module shall be in the EM-A-IDLE state.

2.2.1.5.3.13.2 Upon receipt of an ADS-emergency-report request with no *positive acknowledgement* parameter:

2.2.1.5.3.13.2.1 If in the EM-A-IDLE state, the air EM module shall:

a) create an ADS-emergency-report-PDU with elements as defined in Table 2.2.1.5-52,

b) pass it to the air LI module, and

c) enter the EM-A-ACTIVE state.

2.2.1.5.3.13.2.2 If in the EM-A-ACTIVE state, the air EM module shall:

a) create an ADS-emergency-report-PDU with elements as defined in Table 2.2.1.5-52,

b) pass it to the air LI module, and

c) remain in the EM-A-ACTIVE state.

Table 2.2.1.5-52

Parameter Name	Derivation of Parameter Value
emergency-report	*emergency report details* parameter value
positive-acknowledgements	not provided

2.2.1.5.3.13.3 Upon receipt of an ADS-emergency-report request with a *positive acknowledgement* parameter:

2.2.1.5.3.13.3.1 If in the EM-A-MODIFY state, the air EM module shall:

a) create an ADS-emergency-report-PDU with elements as defined in Table 2.2.1.5-53,

b) pass it to the air LI module, and

c) enter the EM-A-ACTIVE state.

Table 2.2.1.5-53

Parameter Name	Derivation of Parameter Value
emergency-report	*emergency report details* parameter value
positive-acknowledgements	NULL

2.2.1.5.3.13.4 Upon receipt of an ADS-cancel-emergency request:

2.2.1.5.3.13.4.1 If in the EM-A-ACTIVE state, the air EM module shall:

a) create an ADS-cancel-emergency-PDU,

b) pass it to the air LI module,

c) start the t-EM-3 timer, and

d) enter the EM-A-CANCEL state.

2.2.1.5.3.13.5 Upon receipt of an ADS-modify-emergency-contract response:

2.2.1.5.3.13.5.1 If in the EM-A-MODIFY state, the air EM module shall:

a) create an ADS-negative-acknowledgement-PDU with elements as defined in Table 2.2.1.5-54,

b) pass it to the air LI module, and

c) enter the EM-A-ACTIVE state.

Table 2.2.1.5-54

PDU Element Name	Derivation of Element Value
request-type	modify-emergency-contract

reason	cannot-meet-reporting-rate

2.2.1.5.3.13.6 Upon receipt of an ADS-cancel-emergency-acknowledgement-PDU:

2.2.1.5.3.13.6.1 If in the EM-A-CANCEL state, the air EM module shall:

a) stop the t-EM-3 timer, and

b) enter the EM-A-IDLE state.

2.2.1.5.3.13.7 Upon receipt of an ADS-modify-emergency-contract-PDU:

2.2.1.5.3.13.7.1 If in the EM-A-ACTIVE state, the air EM module shall:

a) request the HI module to invoke ADS-modify-emergency-contract indication, and

b) enter the EM-A-MODIFY state.

2.2.1.5.3.13.7.2 If in the EM-A-CANCEL state, the air EM module:

a) shall remain in the EM-A-CANCEL state.

2.2.1.5.3.13.8 Upon expiry of the t-EM-3, the air EM module shall:

a) request the air AB module to abort with reason *timer-expiry*, and

b) enter the EM-A-IDLE state.

2.2.1.5.3.13.9 Upon receipt of a request from the air AB or air LI module to stop operation, the air EM module shall:

a) stop any timers, and

b) enter the EM-A-IDLE state.

2.2.1.5.3.14 Ground and Air ADS AB Modules

Note.— All statements in 2.2.1.5.3.14 apply to both the ADS ground AB module and the ADS air AB module.

2.2.1.5.3.14.1 Upon receipt of an ADS-user-abort request, the AB module shall:

a) request the DC, EC, PC and EM modules to stop operation, and

b) request the LI module to invoke D-ABORT with parameter values derived as in Table 2.2.1.5-55.

Table 2.2.1.5-55

Parameter Name	Derivation of Parameter Value
originator	user
user data	not provided

2.2.1.5.3.14.2 Upon receipt of a request to abort from the LI, DC, EC, PC or EM modules, the AB module shall:

a) request the DC, EC, PC and EM modules to stop operation,

b) create an ADS-provider-abort-PDU with elements as defined in Table 2.2.1.5-56,

c) request the LI module to invoke D-ABORT with parameter values derived as in Table 2.2.1.5-57, and

d) request the HI module to invoke ADS-provider-abort with parameter values derived as in Table 2.2.1.5-58.

Table 2.2.1.5-56

PDU Element Name	Derivation of Element Value
AbortReason	Value provided by DC, EC, PC or EM module

Table 2.2.1.5-57

Parameter Name	Derivation of Parameter Value
originator	provider
user data	the ADS-provider-abort-PDU

Table 2.2.1.5-58

Parameter Name	Derivation of Parameter Value
Reason	Value provided by DC, EC, PC or EM module

2.2.1.5.3.14.3 Upon receipt of a D-P-ABORT indication, the AB module shall:

a) request the DC, EC, PC and EM modules to stop operation, and

b) request the HI module to invoke ADS-provider-abort indication with parameter values derived as in Table 2.2.1.5-59.

Table 2.2.1.5-59

Parameter Name	Derivation of Parameter Value
Reason	*communications-service-failure*

2.2.1.5.3.14.4 Upon receipt of a D-ABORT indication with the *originator* parameter value set to the abstract value "user", the AB module shall:

a) request the DC, EC, PC and EM modules to stop operation, and

b) request the HI module to invoke ADS-user-abort indication.

2.2.1.5.3.14.5 Upon receipt of a D-ABORT indication with the *originator* parameter value set to the abstract value "provider", the AB module shall:

a) request the DC, EC, PC and EM modules to stop operation, and

b) request the HI module to invoke ADS-provider-abort indication with parameter values derived as in Table 2.2.1.5-60.

Table 2.2.1.5-60

Parameter Name	Derivation of Parameter Value
Reason	value of the *user data* parameter

2.2.1.5.3.15 Ground ADS LI Module

Note.— The states defined for the ground ADS LI module are the following:

a) LI-G-IDLE

b) LI-G-START

c) LI-G-ACTIVE

d) LI-G-END

2.2.1.5.3.15.1 On initiation, the ground LI module shall be in the LI-G-IDLE state.

2.2.1.5.3.15.2 Upon receipt of an ADS-demand-contract-PDU, an ADS-event-contract-PDU, or an ADS-periodic-contract-PDU from the ground DC, EC or PC modules:

2.2.1.5.3.15.2.1 If in the LI-G-IDLE state, the ground LI module shall:

a) Invoke D-START request using parameter values as shown in Table 2.2.1.5-61, and

b) enter the LI-G-START state.

Table 2.2.1.5-61: D-START request parameter values

D-START parameter	Source
Called peer Id	*Aircraft identifier* parameter value from contract request
Calling peer Id	*ICAO facility designation* parameter value from contract request
DS-user version number	Not used
Security requirements	Not used
Quality of service	Routing class:ATSC, with value from *Class of communication service* parameter value from contract request Priority: High priority flight safety messages RER: Low
User data	The contract PDU passed to LI

2.2.1.5.3.15.2.2 If in the LI-G-ACTIVE state, the ground LI module shall:

a) Invoke D-DATA request with the PDU as the *user data* parameter value, and

b) remain in the LI-G-ACTIVE state.

2.2.1.5.3.15.3 Upon receipt of an ADS-cancel-all-contracts request :

2.2.1.5.3.15.3.1 If in the LI-G-ACTIVE state, the ground LI module shall:

a) Invoke D-END req with ADS-cancel-all-contracts-PDU in the user data,

b) start the t-LI-1 timer, and

c) enter the LI-G-END state.

2.2.1.5.3.15.4 Upon receipt of an ADS-cancel-contract-PDU, ADS-modify-emergency-contract-PDU, or ADS-cancel-emergency-acknowledgement-PDU from the ground DC, EC, PC or EM modules:

2.2.1.5.3.15.4.1 If in the LI-G-ACTIVE state, the ground LI module shall:

a) invoke D-DATA req with PDU in the user data,

b) if the DC module is in the DC-G-IDLE state, and the EC module is in the EC-G-IDLE state, and the PC module is in the PC-G-IDLE state, and the ground EM module is in the EM-G-IDLE state, then:

1) invoke D-END req with no user data,

2) start the t-LI-1 timer, and

3) enter the LI-G-END state; or

c) otherwise, remain in the LI-G-ACTIVE state.

2.2.1.5.3.15.4.2 If in the LI-G-END state, the ground LI module shall remain in the LI-G-END state.

2.2.1.5.3.15.5 Upon receipt of request to invoke D-ABORT from the ground AB module:

2.2.1.5.3.15.5.1 If in the LI-G-START state, the LI-G-ACTIVE state or the LI-G-END state, the ground LI module shall:

a) Invoke D-ABORT request with the parameter values supplied,

b) Start the t-LI-1 timer, and

c) enter the LI-G-IDLE state.

2.2.1.5.3.15.5.2 If in the LI-G-IDLE state, the ground LI module shall ignore the request.

2.2.1.5.3.15.6 Upon receipt of a D-START confirmation with the *Result* parameter value containing the abstract value accepted:

2.2.1.5.3.15.6.1 If in the LI-G-START state, the ground LI module shall:

a) pass the user data to the module as defined in Table 2.2.1.5-62,

b) if, after processing the PDU (i.e. the ground HI module has issued the appropriate indication or confirmation), the ground DC module is in the DC-G-IDLE state, and the ground EC module is in the EC-G-IDLE state, and the ground PC module is in the PC-G-IDLE state, and the ground EM module is in the EM-G-IDLE state, then:

1) invoke D-END req with no user data, and

2) start the t-LI-1 timer, and

3) enter the LI-G-END state; or

c) if, after processing the PDU, the ground DC module is not in the DC-G-IDLE state, or the ground EC module is not in the EC-G-IDLE state, or the ground PC module is not in the PC-G-IDLE state, or the ground EM module is not in the EM-G-IDLE state, then:

1) enter the LI-G-ACTIVE state.

2.2.1.5.3.15.7 Upon receipt of a D-DATA indication:

2.2.1.5.3.15.7.1 If in the LI-G-ACTIVE state, the ground LI module shall:

a) pass the user data to the module as defined in Table 2.2.1.5-62,

b) if, after processing the PDU (i.e. the ground HI module has issued the appropriate indication or confirmation), the ground DC module is in the DC-G-IDLE state, and the ground EC module is in the EC-G-IDLE state, and the ground PC module is in the PC-G-IDLE state, and the ground EM module is in the EM-G-IDLE state, then:

1) invoke D-END req with no user data, and

2) start the T-LI-1 timer, and

3) enter the LI-G-END state; or

c) otherwise, remain in the LI-G-ACTIVE state.

2.2.1.5.3.15.7.2 If in the LI-G-END state, the ground LI module shall:

a) pass the user data to the module as defined in Table 2.2.1.5-62, and

b) remain in the LI-G-END state.

Table 2.2.1.5-62: PDU to ground module mapping

PDU	Subfield	Ground Module
ADS-demand-report-PDU		DC
ADS-event-report-PDU		EC
ADS-periodic-report-PDU		PC
ADS-emergency-report-PDU		EM
ADS-cancel-emergency-PDU		EM

PDU	Subfield	Ground Module
ADS-positive-acknowledgement-PDU	RequestType	
	cancel-event-contract	EC
	cancel-periodic-contract	PC
	demand-contract	DC
	event-contract	EC
	modify-emergency-contract	EM
	periodic-contract	PC
ADS-noncompliance-notification-PDU	contract-type	
	demand-contract	DC
	event-contract	EC
	periodic-contract	PC
ADS-negative-acknowledgement-PDU	RequestType	
	cancel-event-contract	EC
	cancel-periodic-contract	PC
	demand-contract	DC
	event-contract	EC
	modify-emergency-contract	EM
	periodic-contract	PC

2.2.1.5.3.15.8 Upon receipt of a D-END confirmation with the *result* parameter value containing the abstract value *accepted*:

2.2.1.5.3.15.8.1 If in the LI-G-END state, the ground LI module shall:

a) stop the t-LI-1 timer,

b) if the user data parameter value contains an ADS-positive-acknowledgement-PDU (cancel-all-contracts), then request the ground HI module to invoke ADS-cancel-all-contracts confirmation,

c) request the DC, EC, PC and EM modules to stop operation, and

d) enter the LI-G-IDLE state.

2.2.1.5.3.15.9 Upon receipt of a D-ABORT indication, the ground LI module shall:

a) pass the D-ABORT indication to the ground AB module, and

b) enter the LI-G-IDLE state.

2.2.1.5.3.15.10 Upon receipt of a D-P-ABORT indication, the ground LI module shall:

a) pass the D-P-ABORT indication to the ground AB module, and

b) enter the LI-G-IDLE state.

2.2.1.5.3.15.11 Upon expiry of the t-LI-1 timer, the ground LI module shall:

a) request the air AB module to abort with reason *timer-expiry*, and

b) remain in the same state.

2.2.1.5.3.16 Air ADS LI Module

Note.— The states defined for the air ADS LI module are the following:

a) LI-A-IDLE

b) LI-A-START

c) LI-A-ACTIVE

2.2.1.5.3.16.1 On initiation, the air LI module shall be in the LI-A-IDLE state.

2.2.1.5.3.16.2 Upon receipt of an ADS-demand-report-PDU, or an ADS-event-report-PDU, or an ADS-periodic-report-PDU, or an ADS-positive-acknowledgement-PDU, or an ADS-negative-acknowledgement-PDU, or an ADS-noncompliance-notification-PDU from the air DC, EC, PC or EM modules:

2.2.1.5.3.16.2.1 If in the LI-A-START state, the air LI module shall:

a) Invoke D-START response using parameter values as shown in Table 2.2.1.5-63, and

b) enter the LI-A-ACTIVE state.

Table 2.2.1.5-63: D-START response parameter values

D-START parameter	Source
DS-user version number	Not used
Security requirements	Not used
Quality of service	Not used
Result	Accepted
User data	The PDU passed to the air LI module

2.2.1.5.3.16.2.2 If in the LI-A-ACTIVE state, the air LI module shall:

a) Invoke D-DATA request using the PDU as the *user data* parameter value, and

b) remain in the LI-A-ACTIVE state.

2.2.1.5.3.16.3 Upon receipt of an ADS-cancel-emergency-PDU, or an ADS-emergency-report-PDU from the EM module:

2.2.1.5.3.16.3.1 If in the LI-A-ACTIVE state, the air LI module shall:

a) Invoke D-DATA request using the PDU as the *user data* parameter value, and

b) remain in the LI-A-ACTIVE state.

2.2.1.5.3.16.4 Upon receipt of a request to invoke D-ABORT from the ground AB module:

2.2.1.5.3.16.4.1 If in the LI-A-START or LI-A-ACTIVE state, the air LI module shall:

a) Invoke D-ABORT request with the parameter values supplied, and

b) enter the LI-G-IDLE state.

2.2.1.5.3.16.5 Upon receipt of a D-START indication:

2.2.1.5.3.16.5.1 If in the LI-A-IDLE state, and the application service priority parameter value is "high priority flight safety messages", and the RER quality of service parameter is the abstract value "low", the air LI module shall:

a) pass the user data and the calling peer id parameter value to the module as defined in Table2.2.1.5-64, and

b) enter LI-A-START state.

2.2.1.5.3.16.6 Upon receipt of a D-DATA indication:

2.2.1.5.3.16.6.1 If in the LI-A-ACTIVE state, the air LI module shall:

a) pass the user data to the module as defined in Table 2.2.1.5-64, and

b) remain in the LI-A-ACTIVE state.

Table 2.2.1.5-64: PDU to air module mapping

PDU type	Sub-element	Air Module
ADS-demand-contract-PDU		DC
ADS-event-contract-PDU		EC
ADS-periodic-contract-PDU		PC
ADS-modify-emergency-contract-PDU		EM
ADS-cancel-emergency-acknowledgement-PDU		EM
ADS-cancel-all-contracts-PDU		HI
ADS-cancel-contract-PDU	CancelContract	
	event-contract	EC
	periodic-contract	PC

2.2.1.5.3.16.7 Upon receipt of a D-END indication:

2.2.1.5.3.16.7.1 If in the LI-A-ACTIVE state:

2.2.1.5.3.16.7.1.1 If the user data parameter value contains an ADS-cancel-all-contracts-PDU, the air LI module shall:

a) pass it to the air HI module,

b) invoke D-END response with the *Result* parameter value set to *accepted* and ADS-positive-acknowledgement-PDU (cancel-all-contracts) in the *user data*,

c) request the DC, EC, PC and EM modules to stop operation, and

d) enter LI-A-IDLE state.

2.2.1.5.3.16.7.1.2 If there is no user data, the air LI module shall:

a) invoke D-END response with the *result* parameter value set to *accepted* and no *user data*, and

b) enter LI-A-IDLE state.

2.2.1.5.3.16.8 Upon receipt of a D-ABORT indication, the air LI module shall:

a) stop any timer, and

b) pass the D-ABORT indication to the air AB module, and

c) enter the LI-A-IDLE state.

2.2.1.5.3.16.9 Upon receipt of a D-P-ABORT indication, the air LI module shall:

a) stop any timer, and

b) pass the D-P-ABORT indication to the air AB module, and

c) enter the LI-A-IDLE state.

2.2.1.5.4 Exception Handling

2.2.1.5.4.1 Timer Expires

2.2.1.5.4.1.1 When any of the timers in any of the modules stated in 2.2.1.5.2 reaches its maximum time, the module shall request the air or ground AB module to abort with reason *timer-expiry*.

2.2.1.5.4.2 Unrecoverable System Error

2.2.1.5.4.2.1 **Recommendation**.—*When any module has an unrecoverable system error, the module should request the air or ground AB module to abort with reason unrecoverable-system-error*.

2.2.1.5.4.3 Invalid PDU

2.2.1.5.4.3.1 When the user data parameter value of a D-START indication is a valid APDU and is not an ADS-demand-contract-PDU, an ADS-event-contract-PDU or an ADS-periodic-contract-PDU, the air LI module shall request the AB module to abort with reason invalid-PDU.

2.2.1.5.4.3.2 When the user data parameter value of a D-START confirmation is a valid APDU and is not an ADS-demand-report-PDU, an ADS-event-report-PDU, an ADS-periodic-report-PDU, an ADS-positive-acknowledgement-PDU, and ADS-negative-acknowledgement-PDU or an ADS-noncompliance-notification-PDU the ground LI module shall request the AB module to abort with reason *invalid-PDU*.

2.2.1.5.4.3.3 When the user data parameter value of a D-DATA indication is a valid APDU and is not an ADS-demand-contract-PDU, an ADS-event-contract-PDU, an ADS-periodic-contract-PDU, an ADS-modify-emergency-contract-PDU, an ADS-cancel-emergency-acknowledgement-PDU, an ADS-cancel-all-contracts-PDU or an ADS-cancel-contract-PDU, the air LI module shall request the AB module to abort with reason *invalid-PDU*.

2.2.1.5.4.3.4 When the user data parameter value of a D-DATA indication is a valid APDU and is not an ADS-demand-report-PDU, an ADS-event-report-PDU, an ADS-periodic-report-PDU, an ADS-positive-acknowledgement-PDU, and ADS-negative-acknowledgement-PDU, an ADS-noncompliance-notification-PDU, an ADS-emergency-report-PDU or an ADS-cancel-emergency-PDU the ground LI module shall request the AB module to abort with reason *invalid-PDU*.

2.2.1.5.4.3.5 When the user data parameter value of a D-END indication is present, but does not contain an ADS-cancel-all-contracts-PDU, the air LI module shall request the AB module to abort with reason *invalid-PDU*.

2.2.1.5.4.3.6 When the user data parameter value of a D-ABORT indication is a valid APDU and is not an ADS-provider-abort-PDU, the air LI module or the ground LI module shall request the AB module to abort with reason *invalid-PDU*.

2.2.1.5.4.3.7 When the user data parameter value of a D-END confirmation is present, but does not contain an ADS-positive-acknowledgement-PDU (cancel-all-contracts), the ground LI module shall request the AB module to abort with reason *invalid-PDU*.

2.2.1.5.4.4 Sequence Error

2.2.1.5.4.4.1 When a PDU is passed to a module for which there are no instructions in 2.2.1.5.3 (i.e. the PDU has arrived out of sequence), the air or ground AB module shall be requested to abort with reason *sequence-error*.

2.2.1.5.4.5 D-START Rejection

2.2.1.5.4.5.1 Upon receipt of a D-START confirmation with the *result* parameter value containing the abstract value rejected (transient) or rejected (permanent), and the *reject source* parameter value containing the abstract value DS user, the ground LI module shall:

a) request the ground AB module to abort with reason *sequence-error*; and

b) enter the LI-G-IDLE state.

2.2.1.5.4.5.2 Upon receipt of a D-START confirmation with the *result* parameter value containing the abstract value rejected (transient) or rejected (permanent), and the *reject source* parameter value containing the abstract value DS provider, the ground LI module shall:

a) request the ground AB module to abort with reason *sequence-error*; and

b) enter the LI-G-IDLE state.

2.2.1.5.4.6 D-END Rejection

2.2.1.5.4.6.1 Upon receipt of a D-END confirmation with the *result* parameter value containing the abstract value rejected, the air or ground AB module shall be requested to abort with reason *dialogue-end-not-accepted*.

2.2.1.5.4.7 Decoding Error

2.2.1.5.4.7.1 When the air LI module or the ground LI module fails to decode an APDU, the LI module shall request the AB module to abort with reason *decoding-error*.

2.2.1.5.4.7.2 Upon receipt of a D-START indication, a D-START confirmation or a D-DATA indication with no user data, the air or ground AB module shall be requested to abort with reason *decoding-error*.

2.2.1.5.4.8 Invalid QOS

2.2.1.5.4.8.1 Upon receipt of a D-START indication with the application service priority parameter set to a value other than the abstract value "high priority flight safety messages", or the RER quality of service parameter set to a value other than the abstract value "low", the air LI module shall request the air AB module to abort with reason *invalid-qos-parameter*.

2.2.1.5.5 ADS-ASE State Tables

2.2.1.5.5.1 Priority

2.2.1.5.5.1.1 If the state tables for the ADS-air-ASE and the ADS-ground-ASE shown below conflict with textual statements made elsewhere in this document, the textual statements shall take precedence.

Note 1.— In the following state tables, the statement "cannot occur" means that if the implementation conforms to the SARPs, it is impossible for this event to occur. If the event does occur, this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE aborts with the error "internal system error".

Note 2.— In the following state tables, the statement "not permitted" means that the implementation must prevent this event from occurring through some local means. If the event does occur this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE performs a local rejection of the request rather than aborting the dialogue.

Table 2.2.1.5-65: ADS ground DC module state table

State ⇒ *Event* ⇓	*DC-G-IDLE* (Initial State)	*DC-G-PENDING*	*DC-G-ACTIVE*
Primitive Requests and Responses			
ADS-demand-contract req	Send ADS-demand-contract-P DU Start t-DC-1 *DC-G-PENDING*	Not permitted	Not permitted
ADS downlink PDUs			
ADS-demand-report-PDU (with positive acknowledgement)	request AB to abort	Stop t-DC-1 ADS-report ind *DC-G-IDLE*	request AB to abort
ADS-demand-report-PDU	request AB to abort	request AB to abort	stop t-DC-2 ADS-report ind *DC-G-IDLE*
ADS-negative-acknowledgement-PDU (demand contract)	request AB to abort	stop t-DC-1 ADS-demand-contract cnf *DC-G-IDLE*	request AB to abort
ADS-noncompliance-notification-PDU (demand-ncn)	request AB to abort	stop t-DC-1 ADS-demand-contract cnf start t-DC-2 *DC-G-ACTIVE*	request AB to abort
Requests from other modules			
Request to stop operation	DC-G-IDLE	stop t-DC-1 *DC-G-IDLE*	stop t-DC-2 *DC-G-IDLE*
Timer expiry			
t-DC-1	cannot occur	request AB to abort	cannot occur
t-DC-2	cannot occur	cannot occur	request AB to abort

Table 2.2.1.5-66: ADS air DC module state table

State ⇒ *Event* ⇓	*DC-A-IDLE* (Initial State)	*DC-A-PENDING*	*DC-A-ACTIVE*
Primitive Requests and Responses			
ADS-demand-contract rsp (negative acknowledgement)	Not permitted	Send ADS-negative-acknowledgement-PDU *DC-A-IDLE*	Not permitted
ADS-demand-contract rsp (noncompliance notification)	Not permitted	Send ADS-noncompliance-notification-PDU *DC-A-ACTIVE*	Not permitted
ADS-report req (demand contract with positive acknowledgement)	Not permitted	Send ADS-demand-report-PDU *DC-A-IDLE*	Not permitted
ADS-report req (demand contract)	Not permitted	Not permitted	Send ADS-demand-report-PDU DC-A-IDLE
Requests from other modules			
Requests to stop operation	*DC-A-IDLE*	*DC-A-IDLE*	*DC-A-IDLE*
ADS uplink PDUs			
ADS-demand-contract-PD U	ADS-demand-contract ind *DC-A-PENDING*	request AB to abort	request AB to abort

Table 2.2.1.5-67: ADS ground EC module state table

State ⇒ *Event* ⇓	*EC-G-IDLE* (Initial State)	*EC-G-START-PENDING*	*EC-G-ACTIVE*	*EC-G-PENDING*	*EC-G-CANCEL*
Primitive Requests and Responses					
ADS-event-contract req	Send ADS-event-contract -PDU Start t-EC-1 *EC-G-START-PENDING*	Not permitted	Send ADS-event-contract -PDU Start t-EC-1 *EC-G-PENDING*	Not permitted	Not permitted
ADS-cancel req (event contract)	Not permitted	Not permitted	Send ADS-cancel-PDU Start t-EC-2 *EC-G-CANCEL*	Not permitted	Not permitted
ADS Aircraft PDUs					
ADS-event-report-PDU (with positive acknowledgement)	request AB to abort	Stop t-EC-1 ADS-report ind *EC-G-ACTIVE*	request AB to abort	Stop t-EC-1 ADS-report ind *EC-G-ACTIVE*	request AB to abort
ADS-event-report-PDU	request AB to abort	request AB to abort	ADS-report ind *EC-G-ACTIVE*	ADS-report ind *EC-G-PENDING*	ADS-report ind *EC-G-CANCEL*
ADS-positive-acknowledgement-PDU (event-contract)	request AB to abort	stop t-EC-1 ADS-event-contract cnf *EC-G-ACTIVE*	request AB to abort	stop t-EC-1 ADS-event-contract cnf *EC-G-ACTIVE*	request AB to abort
ADS-positive-acknowledgement-PDU (cancel-contract-event)	request AB to abort	request AB to abort	request AB to abort	request AB to abort	Stop t-EC-2 ADS-cancel-contract cnf *EC-G-IDLE*
ADS-negative-acknowledgement-PDU (event-contract)	request AB to abort	stop t-EC-1 ADS-event-contract cnf *EC-G-IDLE*	request AB to abort	stop t-EC-1 ADS-event-contract cnf *EC-G-ACTIVE*	request AB to abort
ADS-noncompliance-notification-PDU (event-ncn)	request AB to abort	stop t-EC-1 ADS-event-contract cnf *EC-G-ACTIVE*	request AB to abort	stop t-EC-1 ADS-event-contract cnf *EC-G-ACTIVE*	request AB to abort
Requests from other modules					
Requests to stop operation	*EC-G-IDLE*	stop t-EC-1 *EC-G-IDLE*	*EC-G-IDLE*	stop t-EC-1 *EC-G-IDLE*	stop t-EC-2 *EC-G-IDLE*
Timer expiry					
t-EC-1	cannot occur	request AB to abort	cannot occur	request AB to abort	cannot occur
t-EC-2	cannot occur	cannot occur	cannot occur	cannot occur	request AB to abort

Table 2.2.1.5-68: ADS air EC module state table

State ⇒ *Event* ⇓	*EC-A-IDLE* (Initial State)	*EC-A-PENDING*	*EC-A-ACTIVE*	*EC-A-ACTIVE-PENDING*
Primitive Requests and Responses				
ADS-event-contract rsp (positive acknowledgement or noncompliance notification)	Not permitted	Send ADS-positive-acknowledgement-PDU or ADS-noncompliance-notification-PDU *EC-A-ACTIVE*	Not permitted	Send ADS-positive-acknowledgement-PDU or ADS-noncompliance-notification-PDU *EC-A-ACTIVE*
ADS-event-contract rsp (negative acknowldegement)	Not permitted	Send ADS-negative-acknowledgement-PDU *EC-A-IDLE*	Not permitted	Send ADS-negative-acknowledgement-PDU *EC-A-ACTIVE*
ADS-report req (with positive acknowledgement) (event contract)	Not permitted	Send ADS-event-report-PDU *EC-A-ACTIVE*	Not permitted	Send ADS-event-report-PDU EC-A-ACTIVE
ADS-report req (event contract)	Not permitted	Not permitted	Send ADS-event-report-PDU *EC-A-ACTIVE*	Not permitted
Requests from other modules				
Requests to stop operation	*EC-A-IDLE*	*EC-A-IDLE*	*EC-A-IDLE*	*EC-A-IDLE*
ADS Ground PDUs				
ADS-event-contract-PDU	ADS-event-contract ind *EC-A-PENDING*	request AB to abort	ADS-event-contract ind *EC-A-ACTIVE-PENDING*	request AB to abort
ADS-cancel-PDU (event contract)	request AB to abort	request AB to abort	ADS-cancel ind Send ADS-positive-acknowledgement *EC-A-IDLE*	request AB to abort

Table 2.2.1.5-69: ADS-periodic-contract ground based state table

State ⇒ *Event* ⇓	*PC-G-IDLE* (Initial State)	*PC-G-START-PENDING*	*PC-G-ACTIVE*	*PC-G-PENDING*	*PC-G-CANCEL*
Primitive Requests and Responses					
ADS-periodic-contract req	Send ADS-periodic-contract -PDU Start t-PC-1 *PC-G-START-PENDING*	Not permitted	If emergency=FALSE then Stop t-PC-2 Send ADS-periodic-contract -PDU Start t-PC-1 *PC-G-PENDING*	Not permitted	Not permitted
ADS-cancel req	Not permitted	Not permitted	Stop t-PC-2 Send ADS-cancel-PDU Start t-PC-3 *PC-G-CANCEL*	Not permitted	Not permitted
ADS Aircraft PDUs					
ADS-periodic-report-PDU (with positive acknowledgement)	request AB to abort	Stop t-PC-1 ADS-report ind If emergency=FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort	Stop t-PC-1 ADS-report ind If emergency= FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort
ADS-periodic-report-PDU (with no positive acknowledgement)	request AB to abort	request AB to abort	If emergency=FALSE then Stop t-PC-2 ADS-report ind If emergency=FALSE then Start t-PC-2 *PC-G-ACTIVE*	ADS-report ind *PC-G-PENDING*	ADS-report ind *PC-G-CANCEL*
ADS-positive-acknowledgement-PDU (periodic-contract)	request AB to abort	stop t-PC-1 ADS-periodic-contract cnf If emergency=FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort	stop t-PC-1 ADS-periodic-contract cnf If emergency= FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort
ADS-positive-acknowledgement-PDU (cancel-contract - periodic)	request AB to abort	request AB to abort	request AB to abort	request AB to abort	Stop t-PC-3 ADS-cancel-contract cnf *PC-G-IDLE*

State ⇒ *Event* ⇓	*PC-G-IDLE* (Initial State)	*PC-G-START-PENDING*	*PC-G-ACTIVE*	*PC-G-PENDING*	*PC-G-CANCEL*
ADS-negative-acknowledgement-PDU (periodic-contract)	request AB to abort	stop t-PC-1 ADS-periodic-contract cnf *PC-G-IDLE*	request AB to abort	stop t-PC-1 ADS-periodic-contract cnf If emergency= FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort
ADS-noncompliance-notification-PDU	request AB to abort	stop t-PC-1 ADS-periodic-contract cnf If emergency=FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort	stop t-PC-1 ADS-periodic-contract cnf If emergency= FALSE then Start t-PC-2 *PC-G-ACTIVE*	request AB to abort
Requests from other modules					
Request to stop operation	set emergency=FALSE *PC-G-IDLE*	stop t-PC-1 set emergency=FALSE *PC-G-IDLE*	If emergency=FALSE then stop t-PC-2 set emergency=FALSE *PC-G-IDLE*	stop t-PC-1 set emergency= FALSE *PC-G-IDLE*	stop t-PC-3 set emergency=FALSE *PC-G-IDLE*
Request to suspend periodic contract	set emergency=TRUE	set emergency=TRUE	set emergency=TRUE stop t-PC-2	set emergency=TRUE	set emergency=TRUE
Request to reinstate periodic contract	set emergency=FALSE	set emergency=FALSE	set emergency=FALSE start t-PC-2	set emergency= FALSE	set emergency=FALSE
Timer expiry					
t-PC-1	cannot occur	request AB to abort	cannot occur	request AB to abort	cannot occur
t-PC-2	cannot occur	cannot occur	request AB to abort	cannot occur	cannot occur
t-PC-3	cannot occur	cannot occur	cannot occur	cannot occur	request AB to abort

Table 2.2.1.5.70: ADS air PC module state table

State ⇒ *Event* ⇓	*PC-A-IDLE* (Initial State)	*PC-A-PENDING*	*PC-A-ACTIVE*	*PC-A-ACTIVE-PENDING*
Primitive Requests and Responses				
ADS-periodic-contract rsp (positive acknowledgement or noncompliance notification)	Not permitted	Send ADS-positive-acknowledgement-PDU or ADS-noncompliance-notification-PDU *PC-A-ACTIVE*	Not permitted	Send ADS-positive-acknowledgement-PDU or ADS-noncompliance-notification-PDU *PC-A-ACTIVE*
ADS-periodic-contract rsp (negative acknowledgement)	Not permitted	Send ADS-negative-acknowledgement-PDU *PC-A-IDLE*	Not permitted	Send ADS-negative-acknowledgement-PDU *PC-A-ACTIVE*
ADS-report req - with positive acknowledgement (periodic contract)	Not permitted	Send ADS-periodic-report-PDU *PC-A-ACTIVE*	Not permitted	Send ADS-periodic-report-PDU *PC-A-ACTIVE*
ADS-report req (periodic contract)	Not permitted	Not permitted	Send ADS-periodic-report-PDU *PC-A-ACTIVE*	Not permitted
Requests from other modules				
Requests to stop operation	*PC-A-IDLE*	*PC-A-IDLE*	*PC-A-IDLE*	*PC-A-IDLE*
ADS uplink PDUs				
ADS-periodic-contract-PDU	ADS-periodic-contract ind *PC-A-PENDING*	request AB to abort	ADS-periodic-contract ind *PC-A-ACTIVE-PENDING*	request AB to abort
ADS-cancel-PDU (periodic contract)	request AB to abort	request AB to abort	ADS-cancel ind Send ADS-positive-acknowledgement *PC-A-IDLE*	request AB to abort

Table 2.2.1.5.71: ADS ground EM module state table

State ⇒ *Event* ⇓	*EM-G-IDLE*	*EM-G-ACTIVE*	*EM-G-MODIFY*
Primitive Requests and Responses			
ADS-modify-emergency-contract req	Not permitted	Stop t-EM-1 Send ADS-modify-emergency-contract-PDU Start t-EM-2 *EM-G-MODIFY*	Not permitted
ADS Aircraft PDUs			
ADS-emergency-report-PDU (with positive acknowledgement)	request AB to abort	request AB to abort	Stop t-EM-2 ADS-emergency-report ind Start t-EM-1 *EM-G-ACTIVE*
ADS-emergency-report-PDU	Suspend periodic contract ADS-emergency-report ind Start t-EM-1 *EM-G-ACTIVE*	Stop t-EM-1 ADS-emergency-report ind Start t-EM-1 *EM-G-ACTIVE*	ADS-emergency-report ind *EM-G-MODIFY*
ADS-cancel-emergency-PDU	request AB to abort	Stop t-EM-1 ADS-cancel-emergency ind Send ADS-cancel-emergency-acknowledgement-PDU Re-instate periodic contracts *EM-G-IDLE*	Stop t-EM-2 ADS-cancel-emergency ind Send ADS-cancel-emergency-acknowledgement-PDU Re-instate periodic contracts *EM-G-IDLE*
ADS-negative-acknowledgement-PDU (modify-emergency-contract)	request AB to abort	request AB to abort	Stop t-EM-2 ADS-modify-emergency-contract cnf Start t-EM-1 *EM-G-ACTIVE*
Requests from other modules			
Requests to stop operation	*EM-G-IDLE*	stop t-EM-1 *EM-G-IDLE*	stop t-EM-2 *EM-G-IDLE*
Timer expiry			
t-EM-1	cannot occur	request AB to abort	cannot occur
t-EM-2	cannot occur	cannot occur	request AB to abort

Table 2.2.1.5.72: ADS air EM module state table

State ⇒ *Event* ⇓	*EM-A-IDLE*	*EM-A-ACTIVE*	*EM-A-MODIFY*	*EM-A-CANCEL*
Primitive Requests and Responses				
ADS-emergency-report req	Send ADS-emergency-report-PDU *EM-A-ACTIVE*	Send ADS-emergency-report-PDU *EM-A-ACTIVE*	Not permitted	Not permitted
ADS-emergency-report req (with positive acknowledgement)	Not permitted	Not permitted	Send ADS-emergency-report-PDU *EM-A-ACTIVE*	Not permitted
ADS-cancel-emergency req	Not permitted	Send ADS-cancel-emergency-PDU Start t-EM-3 *EM-A-CANCEL*	Not permitted	Not permitted
ADS-modify-emergency-contract rsp	Not permitted	Not permitted	Send ADS-negative-acknowledgement-PDU *EM-A-ACTIVE*	Not permitted
ADS Ground PDUs				
ADS-cancel-emergency-acknowledgement-PDU	request AB to abort	request AB to abort	request AB to abort	Stop t-EM-3 *EM-A-IDLE*
ADS-modify-emergency-contract-PDU	request AB to abort	ADS-modify-emergency-contract ind EM-A-MODIFY	request AB to abort	*EM-A-CANCEL*
Requests from other modules				
Requests to stop operation	*EM-A-IDLE*	*EM-A-IDLE*	*EM-A-IDLE*	stop t-EM-3 *EM-A-IDLE*
Timer expiry				
t-EM-3	cannot occur	cannot occur	cannot occur	request AB to abort

Table 2.2.1.5-73: Ground ADS LI module state table

State ⇒ *Event* ⇓	*LI-G-IDLE* (Initial State)	*LI-G-START*	*LI-G-ACTIVE*	*LI-G-END*
Data and requests passed from other modules				
ADS-demand-contract-request-PDU, ADS-event-contract-request-PDU, or ADS-periodic-contract-request-PDU,	D-START req *LI-G-START*	Not permitted	D-DATA req *LI-G-ACTIVE*	Not permitted
ADS-cancel-all-contracts req	Not permitted	Not permitted	D-END req *LI-G-END*	Not permitted
ADS-cancel-contract-PDU, ADS-modify-emergency-contract-PDU or ADS-cancel-emergency-acknowledgement-PDU	Not permitted	Not permitted	D-DATA req [1]	Not permitted
ADS-provider-abort-PDU	*LI-G-IDLE*	D-ABORT req *LI-G-IDLE*	D-ABORT req *LI-G-IDLE*	D-ABORT req *LI-G-IDLE*
ADS-forward-contract-response-PDU	Not permitted	Not permitted	Not permitted	Not permitted
Primitive Indications and Confirmations				
D-START ind	pass user data to appropriate module *LI-G-START-R*	cannot occur	cannot occur	cannot occur
D-START cnf	cannot occur	pass user data to appropriate module [1]	cannot occur	cannot occur
D-DATA ind	cannot occur	cannot occur	pass user data to appropriate module [1]	pass user data to appropriate module *LI-G-END*
D-END-ind	cannot occur	cannot occur	if ADS-end-forward-service-PDU, pass to HI module D-END rsp *LI-G-IDLE*	cannot occur
D-END cnf	cannot occur	cannot occur	cannot occur	*LI-G-IDLE*
D-ABORT ind or D-P-ABORT ind	cannot occur	pass to AB module *LI-G-IDLE*	pass to AB module *LI-G-IDLE*	pass to AB module *LI-G-IDLE*

[1] If DC, EC, PC, and EM modules are all in their idle state then
Invoke D-END req with no user data
LI-G-END
else
LI-G-ACTIVE

Table 2.2.1.5.74: Air ADS LI module state table

State ⇒ *Event* ⇓	*LI-A-IDLE* (Initial State)	*LI-A-START*	*LI-A-ACTIVE*
Data and requests passed from other modules			
ADS-demand-report-PDU, ADS-negative-acknowledgement-PDU, ADS-event-report-PDU, ADS-periodic-report-PDU, ADS-positive-acknowledgement-PDU, ADS-noncompliance-notification-PDU	Not permitted	D-START rsp *LI-A-ACTIVE*	D-DATA req *LI-A-ACTIVE*
ADS-cancel-emergency-PDU , ADS-emergency-report-PDU	Not permitted	Not permitted	D-DATA req *LI-A-ACTIVE*
Primitive Indications and Confirmations			
D-START ind	pass to appropriate module *LI-G-START*	cannot occur	cannot occur
D-DATA ind	cannot occur	cannot occur	pass user data to appropriate module *LI-A-ACTIVE*
D-END ind	cannot occur	cannot occur	if ADS-cancel-all-contracts-PDU, pass to HI module D-END rsp *LI-A-IDLE*
D-ABORT ind or D-P-ABORT ind	cannot occur	pass to AB module *LI-A-IDLE*	pass to AB module *LI-A-IDLE*

2.2.1.6 Communication Requirements

2.2.1.6.1 Encoding Rules

2.2.1.6.1.1 The ADS application shall use PER as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in 2.2.1.4.

2.2.1.6.2 Dialogue Service Requirements

2.2.1.6.2.1 Primitive Requirements

2.2.1.6.2.1.1 Where dialogue service primitives, that is D-START, D-END, D-ABORT, D-P-ABORT and D-DATA are described as being invoked in 2.2.1.5, the ADS-ground-ASE and the ADS-air-ASE shall exhibit external behaviour consistent with the dialogue service, as described in 4.2, having been implemented and its primitives invoked.

2.2.1.6.2.2 Quality of Service Requirements

2.2.1.6.2.2.1 The application service priority for ADS shall have the abstract value of "high priority flight safety messages".

2.2.1.6.2.2.2 The RER quality of service parameter of the D-START request shall be set to the abstract value of "low".

2.2.1.6.2.2.3 The ADS-ASE shall map the class of communication service abstract values to the ATSC routing class abstract value part of the D-START QOS parameter as presented in Table 2.2.1.6-1.

Table 2.2.1.6-1. Mapping between class of communication and routing class abstract values

Class of Communication Abstract Value	Routing Class Abstract Value
A	Traffic follows Class A ATSC route(s)
B	Traffic follows Class B ATSC route(s)
C	Traffic follows Class C ATSC route(s)
D	Traffic follows Class D ATSC route(s)
E	Traffic follows Class E ATSC route(s)
F	Traffic follows Class F ATSC route(s)
G	Traffic follows Class G ATSC route(s)
H	Traffic follows Class H ATSC route(s)

Note.— ATSC values are defined in 1.3.

2.2.1.7 ADS User Requirements

2.2.1.7.1 General

2.2.1.7.1.1 General Requirements

2.2.1.7.1.1.1 The ADS-ground-user shall only establish a demand contract, an event contract or a periodic contract with an ADS-air-user.

2.2.1.7.1.1.2 The ADS-air-user shall invoke ADS-report requests only at the rate specified and containing only the information required to meet the contract as specified in 2.2.1.7.

2.2.1.7.1.2 General Parameter Requirements

Note 1.— When an ADS-ground-user invokes ADS-demand-contract request, ADS-event-contract request, ADS-periodic-contract request or ADS-forward-contract request and requires a particular class of communication service, it provides the class of communication service parameter.

Note 2.— When an ADS-ground-user invokes ADS-demand-contract request, ADS-event-contract request, ADS-periodic-contract request or ADS-forward-contract request, and does not provide the class of communications service parameter, this indicates no routing preference.

Note 3.— When an ADS-ground-user specifies the class of communications service parameter and there is an ADS contract in place, the parameter is ignored.

2.2.1.7.1.2.1 When providing the air speed (as part of the air vector parameter), the ADS-air-user shall:

a) if available, provide Mach number,

b) if available, provide indicated air speed, or

c) if available, provide both Mach number and indicated air speed.

2.2.1.7.1.3 Timing Requirements

2.2.1.7.1.3.1 **Recommendation.—** *When an ADS-air-user or ADS-ground-user receives an indication that requires a response, it should invoke the response within 0.5 seconds.*

2.2.1.7.1.3.2 **Recommendation.—** *When a periodic contract or an emergency contract is in place, the ADS-air-user should invoke ADS-report request or ADS-emergency report request (as described below) within 0.5 second of the reporting interval as measured from the sending of the previous report.*

2.2.1.7.1.4 Error Handling Requirements

2.2.1.7.1.4.1 If the ADS-air-user or ADS-ground-user has an unrecoverable system error, then it shall:

a) cease the operation of all contracts with peer system(s) which are effected by the error, and

b) for each effected peer system, invoke ADS-user-abort request.

2.2.1.7.1.4.2 If the ADS-user receives an ADS-user-abort indication or an ADS-provider-abort indication, then it shall cease operation of all ADS contracts with the peer system to which the indication is related.

2.2.1.7.1.5 Miscellaneous Air User Requirements

2.2.1.7.1.5.1 With the permissible exception of ADS-user-abort and ADS-provider-abort, the ADS-air-user shall respond to indications and confirmations in the order in which they are received.

2.2.1.7.1.5.2 The ADS-air-user shall be capable of supporting contracts from at least four different ATC ground systems at the same time.

Note.— The ADS-air-user may use the class of communications service indicated in the D-START indication quality of service parameter value in order to determine if the ground systems is an ATC ground system. How the ADS-air-user finds out what class of communications service parameter value is used is a local matter.

2.2.1.7.1.5.3 If the ADS-air-user receives an ADS-demand-contract request, or an ADS-event-contract-request or an ADS-periodic-contract-request which exceeds its capacity for supporting ground systems, then it shall:

a) reject the contract with the *reply* parameter set to *negative acknowledgement*,

b) set the *reason* element of *negative acknowledgement* to *maximum-capacity-exceeded*, and

c) include the set of ICAO facility designations of all the ground systems with which it has contracts in the *maximum-capacity-exceeded* element.

2.2.1.7.1.5.4 If the ICAO facility designation parameter is provided in an ADS-demand-contract indication, an ADS-event-contract indication or an ADS-periodic-contract indication, and if this ICAO facility designation is equal to the ICAO facility designations of any other ground system with which the aircraft has one or more contracts, the ADS-air-user shall invoke ADS-user-abort in place of the normal response.

Note.— The intention is that the new connection will be aborted; the existing connection and all the contracts on it will be retained.

2.2.1.7.1.5.5 If, after accepting a contract, the ADS-air-user is unable to provide the information required, either because it is unavailable, invalid or because its validity is uncertain, then:

a) if the information forms part of position, timestamp or FOM, then:

1) the ADS-air-user shall continue to send ADS reports as required with the FOM set to "0"; and

2) if all of the information is again found to be valid and available, the FOM shall be reset to its actual size.

b) if the information is the aircraft-address, the ADS-air-user shall omit the aircraft-address from any ADS-reports or ADS-emergency-reports that require it, and

c) if the information is part of the projected-profile, the ADS-air-user shall omit the projected-profile from any ADS-reports that require it, and

d) if the information is part of the ground-vector, the ADS-air-user shall omit the ground-vector from any ADS-reports or ADS-emergency-reports that require it, and

e) if the information is part of the air-vector, the ADS-air-user shall omit the air-vector from any ADS-reports that require it, and

f) if the information is part of the weather, the ADS-air-user shall omit the weather from any ADS-reports that require it, and

g) if the information is part of the short-term-intent, the ADS-air-user shall omit the short-term-intent from any ADS-reports that require it, and

h) if the information is part of the extended-projected-profile, the ADS-air-user shall omit the extended-projected-profile from any ADS-reports that require it.

Note 1.— If information is not available for more than one optional field, then both are omitted.

Note 2.— The ADS-air-user must be able to detect when information becomes unavailable. The ADS-air-user must be able to detect if the information is invalid, or its validity is uncertain.

Note 3.— The ADS-ground-user will know what information is expected in any ADS-report or ADS-emergency-report. It is therefore able to tell when the information is unavailable or possibly invalid.

2.2.1.7.1.6 Miscellaneous Ground User Requirements

2.2.1.7.1.6.1 With the permissible exception of ADS-user-abort and ADS-provider-abort, the ADS-ground-user shall respond to indications and confirmations in the order in which they are received.

Note.— The ADS-ground-user checks the contents of the ADS reports received, for conformance to the contracts in place, and flags any non-conformance.

2.2.1.7.2 Establishment and operation of a Demand Contract

Note 1.— 2.2.1.7.2 details the actions taken by the ADS-ground-user and the ADS-air-user in the establishment and operation of a demand contract.

Note 2.— When the ADS-ground-user requires to establish a demand contract with the ADS-air-user, it invokes ADS-demand-contract request.

2.2.1.7.2.1 When the ADS-air-user receives an ADS-demand-contract indication, and is not able to accept the contract, the ADS-air-user shall invoke an ADS-demand-contract response with the *reply* parameter set to *negative-acknowledgement*, and the *reason* parameter set to the value indicating the reason that it cannot accept the contract.

2.2.1.7.2.2 When the ADS-air-user receives an ADS-demand-contract indication, and it is able to accept the contract in full, the ADS-air-user shall invoke an ADS-report request including a *positive acknowledgement* parameter.

2.2.1.7.2.3 When the ADS-air-user receives an ADS-demand-contract indication, and it is able to accept the contract, but is not able to supply all the requested information,

2.2.1.7.2.3.1 the ADS-air-user shall:

a) invoke ADS-demand-contract response, with the *reply* parameter set to *noncompliance-notification*, and the *demand-ncn* parameter element containing an indication of the reports that were requested but cannot be provided, and

b) invoke ADS-report request containing the information that it is able to send, with the *positive acknowledgement* parameter absent.

2.2.1.7.2.3.2 **Recommendation.—** *the ADS-air-user should invoke ADS-report request containing the information that it is able to send, with the positive acknowledgement parameter absent, within 0.5 seconds.*

2.2.1.7.2.4 Forming the ADS-report request

2.2.1.7.2.4.1 Subject to the restrictions stated in 2.2.1.7.1.5.5, the ADS-air-user invokes ADS-report request in response to an ADS-demand-contract, and only when requested in the ADS-demand-contract indication and not indicated as being unavailable in an ADS-demand-contract response (where the *reply* parameter was set to *noncompliance-notification*), then the ADS-air-user shall form the *report details* parameter with the following information:

a) *aircraft address*,

b) *projected-profile*,

c) *ground-vector*,

d) *air-vector*,

e) *weather*,

f) *short-term-intent*, and

g) *extended-projected-profile*.

2.2.1.7.2.4.2 When *short-term-intent* is provided it shall cover the time period indicated in *short-term-intent*.

2.2.1.7.2.4.3 When *extended-projected-profile* is provided it shall cover the time period indicated in *time-interval* or the number of way points indicated in *number-of-way-points* of *extended-project-profile*.

2.2.1.7.2.5 When the ADS-air-user invokes ADS-report request in response to an ADS-demand-contract indication, the *contract type* parameter shall be set to *demand-contract*, and the *event type* parameter not provided.

2.2.1.7.3 Establishment and operation of an Event Contract

Note 1.— 2.2.1.7.3 details the actions taken by the ADS-ground-user and the ADS-air-user in the establishment and operation of a event contract.

Note 2.— When the ADS-ground-user requires to establish an event contract with the ADS-air-user, it invokes ADS-event-contract request.

2.2.1.7.3.1 When invoking the ADS-event-contract request, the ADS-ground-user shall specify at least one event type.

2.2.1.7.3.2 When the ADS-air-user receives an ADS-event-contract indication, and it is not able to accept the contract, then the ADS-air-user shall invoke an ADS-event-contract response with the *reply* parameter set to *negative-acknowledgement* and *reason* set to the value indicating the reason that it cannot accept the contract.

Note.— In the event of the new event contract not being accepted, any existing event contract will remain in place.

2.2.1.7.3.3 When the ADS-air-user receives an ADS-event-contract indication, and it is able to accept the contract in full, the ADS-air-user shall:

a) if the terms of the contract require an ADS-report as baseline information, and the ADS-air-user is able to invoke an ADS-report request within 0.5 seconds, then invoke an ADS-report request including a *positive acknowledgement* parameter, or

b) if the terms of the contract do not require an ADS-report as baseline information, or the ADS-air-user is not able to invoke an ADS-report request within 0.5 seconds, then invoke ADS-event-contract response with *reply* set to *positive acknowledgement*.

2.2.1.7.3.4 When the ADS-air-user receives an ADS-event-contract indication, and it is able partially to fulfill the contract, because it is not able to detect some of the events in the contract, then the ADS-air-user shall invoke ADS-event-contract response with the *reply* parameter set to *noncompliance-notification*, and the event-ncn element set to the events that cannot be complied with.

2.2.1.7.3.5 If the ADS-air-user accepts the event contract with a *noncompliance-notification*, or with a *positive-acknowledgement* (either in an ADS-event-contract response or an ADS-report request) then the ADS-air-user shall:

a) cancel any other event contract with that ground system, and

b) if one or more of the following event types are in the ADS-event-contract indication and not present in the noncompliance notification if sent, then invoke ADS-report request with the contract type set to event-report, the event-type set to baseline, and air-vector and ground-vector included in the report details parameter:

1) air-speed-change,

2) ground-speed-change,

3) heading-change,

4) track-angle-change and/or

5) level-change.

Note.— This provides a baseline reference against which possible deviations are compared.

2.2.1.7.3.5.1 Subject to the restrictions stated in 2.2.1.7.1.5.5, *lateral-deviation-change* is provided in the ADS-event-contract *contract details* parameter, and not indicated in the noncompliance notification if sent, then for the duration of the event contract, only while the lateral deviation of the aircraft relative to the active route of flight is more than the value of *lateral-deviation-change*, the ADS-air-user shall invoke ADS-report requests at a rate of once every 60 seconds, including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.2 Subject to the restrictions stated in 2.2.1.7.1.5.5, *vertical-rate-change* is provided in the ADS-event-contract *contract details* parameter with a zero or positive value, and not indicated in the noncompliance notification if sent, then for the duration of the event contract, only when the aircraft's rate of climb is greater than the value of *vertical-rate-change*, the ADS-air-user shall invoke ADS-report requests at a rate of once every 60 seconds, including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.3 Subject to the restrictions stated in 2.2.1.7.1.5.5, *vertical-rate-change* is provided in the ADS-event-contract *contract details* parameter with a negative value, and not indicated in the noncompliance notification if sent, then for the duration of the event contract, only when the aircraft's rate of descent is greater than the absolute value of *vertical-rate-change*, the ADS-air-user shall invoke ADS-report requests at a rate of once every 60 seconds, including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.4 Subject to the restrictions stated in 2.2.1.7.1.5.5, *level threshold* is provided in the ADS-event-contract *contract details* parameter, and not indicated in the noncompliance notification if sent, then for the duration of the event contract, only when the aircraft's level is greater than the value of *ceiling*, or less than the value of *floor*, the ADS-air-user shall invoke ADS-report requests at a rate of once every 60 seconds, including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.5 Subject to the restrictions stated in 2.2.1.7.1.5.5, *way-point-change* is provided in the ADS-event-contract *contract details* parameter, and not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever the aircraft's next way-point changes, the ADS-air-user shall invoke ADS-report request, including the *projected-profile* element in the *report details* parameter.

2.2.1.7.3.5.6 Subject to the restrictions stated in 2.2.1.7.1.5.5, *fom-change* is provided in the ADS-event-contract *contract details* parameter, then for the duration of the event contract, whenever the aircraft's navigational accuracy, navigational system redundancy or airborne collision avoidance system (ACAS) availability changes, the ADS-air-user shall invoke ADS-report request.

2.2.1.7.3.5.7 Subject to the restrictions stated in 2.2.1.7.1.5.5, *extended-projected-profile-change* is provided in the ADS-event-contract *contract details* parameter, and contains the *time-interval* element, and is not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever one or more way-points on the active route of flight within the *time-interval* as measured from the current time changes, the ADS-air-user shall invoke ADS-report request including the *extended-projected-profile* element containing way-points covering the *time-interval* from the current time in the ADS-report request *report details* parameter.

2.2.1.7.3.5.8 Subject to the restrictions stated in 2.2.1.7.1.5.5, *extended-projected-profile-change* is provided in the ADS-event-contract *contract details* parameter, and contains the *number-of-way-points* element, and is not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever one or more way-points on the active route of flight that are in the next *number-of-way-points*, the ADS-air-user shall invoke ADS-report request including the *extended-projected-profile* element containing the next *number-of-way-points*.

2.2.1.7.3.5.9 Subject to the restrictions stated in 2.2.1.7.1.5.5, *air-speed-change* is provided in the ADS-event-contract in the *contract details* parameter, and is not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever the absolute value of the difference between the aircraft's airspeed and the airspeed transmitted in the most recent ADS-report request that contained an air-vector element, is greater than or equal to the value of *air-speed-change*, then the ADS-air-user shall invoke ADS-report request including the *air-vector* element in the *report details* parameter.

2.2.1.7.3.5.10 Subject to the restrictions stated in 2.2.1.7.1.5.5, *ground-speed-change* is provided in the ADS-event-contract in the *contract details* parameter, and is not indicated in the noncompliance notification

if sent, then for the duration of the event contract, whenever the absolute value of the difference between the aircraft's ground speed and the ground speed transmitted in the most recent ADS-report request that contained a *ground-vector* element is greater than or equal to the value of *ground-speed-change*, then the ADS-air-user shall invoke ADS-report request including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.11 Subject to the restrictions stated in 2.2.1.7.1.5.5, *track-angle-change* is provided in the ADS-event-contract in the *contract details* parameter, and is not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever the absolute value of the difference between the aircraft's track angle and the track angle transmitted in the most recent ADS-report request that contained a *ground-vector* element, is greater than or equal to the value of *track-angle-change*, then the ADS-air-user shall invoke ADS-report request including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.12 Subject to the restrictions stated in 2.2.1.7.1.5.5, *level-change* is provided in the ADS-event-contract in the *contract details* parameter, and is not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever the absolute value of the difference between the aircraft's *level* and the level transmitted in the most recent ADS-report request, is greater than or equal to the value of *level-change*, then the ADS-air-user shall invoke ADS-report request including the *ground-vector* element in the *report details* parameter.

2.2.1.7.3.5.13 Subject to the restrictions stated in 2.2.1.7.1.5.5, *heading-change* is provided in the ADS-event-contract *contract details* parameter, and not indicated in the noncompliance notification if sent, then for the duration of the event contract, whenever the aircraft's heading differs negatively or positively from the value transmitted in the previous ADS report containing an air-vector element by an amount exceeding the value of the *heading-change* element specified in the event contract request, then the ADS-air-user shall invoke ADS-report request including the *air-vector* element in the *report details* parameter.

2.2.1.7.3.5.14 If the ability of the aircraft to detect the occurrence of events changes during the event contract to the extent that it may effect the ability of the aircraft to meet the terms of the event contract, the ADS-air-user shall invoke ADS-report request including the *ability-to-detect-events-impaired* element in the *report details* parameter.

Note 1.— If more than one of the events described above occurs at the same time, the ADS-air-user invokes separate ADS-report requests as described above, for each event independently (i.e. the same report cannot be used to report on more than one event, even if the same information is being transmitted.)

Note 2.— Apart from circumstances detailed in 2.2.1., the positive acknowledgement parameter is not present in any ADS-report request made in response to an ADS-event-contract indication.

2.2.1.7.3.6 When the ADS-air-user invokes ADS-report request in response to an ADS-event-contract indication, the *contract type* parameter shall be set to *event-contract*.

2.2.1.7.3.7 When the ADS-air-user invokes ADS-report request in response to an ADS-event-contract indication, the *event type* parameter shall be set to indicate the type of event in the contract that this report is in response to, or to indicate a baseline report.

2.2.1.7.4 Establishment and operation of a Periodic Contract

Note 1.— 2.2.1.7.4 details the actions taken by the ADS-ground-user and the ADS-air-user in the establishment and operation of a periodic contract while no emergency contract exists.

Note 2.— When the ADS-ground-user requires to establish a periodic contract with the ADS-air-user it invokes ADS-periodic-contract request.

2.2.1.7.4.1 When the ADS-air-user receives an ADS-periodic-contract indication, and it is not able to accept the contract, then the ADS-air-user shall invoke an ADS-periodic-contract response with the *reply* parameter set to *negative-acknowledgement* and reason set to the value indicating the *reason* that it cannot accept the contract.

Note.— In the event of the new contract not being accepted, any existing contract will remain in place

2.2.1.7.4.2 When the ADS-air-user receives an ADS-periodic-contract indication, and it is able to accept the contract in full, then:

2.2.1.7.4.2.1 If the ADS-air-user is able to send the first ADS-report request of the contract within 0.5 seconds, then the ADS-air-user shall invoke the first ADS-report request of the contract, including a *positive acknowledgement* parameter.

2.2.1.7.4.2.2 If the ADS-air-user is not able to send the first ADS-report request of the contract within 0.5 seconds, then:

2.2.1.7.4.2.2.1 The ADS-air-user shall:

a) invoke ADS-periodic-contract response with the *reply* parameter set to *positive acknowledgement*, and

b) if no emergency contract exists, send the first ADS-report request of the contract.

2.2.1.7.4.2.2.2 **Recommendation.—** *The ADS-air-user should:*

a) invoke ADS-periodic-contract response with the reply parameter set to positive acknowledgement within 0.5 seconds, and

b) if no emergency contract exists, send the first ADS-report request of the contract within 30 seconds from the receipt of the ADS-periodic-contract request.

2.2.1.7.4.3 When the ADS-air-user receives an ADS-periodic-contract indication, and it is able to supply some of the information required in the contract, but is not able generate all the report elements, or it is not able to meet the requested reporting rate, or both, then the ADS-air-user shall invoke ADS-periodic-contract response with the *reply* parameter set to *noncompliance-notification*, and with *periodic-ncn* set to indicate the reports that cannot be generated and/or that the reporting rate cannot be met.

2.2.1.7.4.4 If the ADS-air-user accepts the periodic contract with an ADS-periodic-contract response with the *Reply* parameter value set to *noncompliance-notification*, or *positive-acknowledgement*, or the ADS-air-user accepts the periodic contract with an ADS-report with the *Positive acknowledgement* parameter present then:

2.2.1.7.4.4.1 The ADS-air-user shall cancel any periodic contract in force with the ground system.

2.2.1.7.4.4.2 If the ADS-air-user accepted the contract with a *noncompliance-notification* that indicated that the reporting rate could not be met, then the ADS-air-user shall set the reporting rate to be 60 seconds.

2.2.1.7.4.4.3 If the ADS-air-user accepted the contract by a means other than a *noncompliance-notification* that indicated that the reporting rate could not be met, then the ADS-air-user shall set the reporting rate to be the *reporting-interval* from the *contract details* parameter of the ADS-periodic-contract indication.

2.2.1.7.4.4.4 The ADS-air-user shall invoke ADS-report requests at the reporting rate, until such time as the contract is cancelled, or suspended due to an emergency.

Note.— If an emergency contract is already in place, the periodic contract will be immediately suspended due to the provisions stated in 2.2.1.7.

2.2.1.7.4.4.5 Subject to the restrictions stated in 2.2.1.7.1.5.5, the ADS-air-user invokes ADS-report request in response to a ADS-periodic-contract indication, then, for each row in Table 2.2.1.7-1 :

a) if the modulus is present in the *contract details* parameter of the ADS-periodic-contract indication;

b) if the ADS-air-user did not accept the contract by means of a *noncompliance-notification* that indicated that it is not able to generate that report element; and

c) if the number of ADS-report requests already invoked in response to this contract is exactly divisible by the value of the modulus parameter.

then the ADS-air-user shall include the *report details* element as indicated in Table 2.2.1.7-1.

Table 2.2.1.7-1: Inclusion of optional report details in ADS-reports

Modulus in the Contract Details Parameter	ADS-report Report Details Element
aircraft address modulus	aircraft address
projected-profile-modulus	projected-profile
ground-vector-modulus	ground-vector
air-vector-modulus	air-vector
weather-modulus	weather

Modulus in the Contract Details Parameter	ADS-report Report Details Element
short-term-intent	short-term-intent
extended-projected-profile-modulus	extended-projected-profile

Note 1.— For example, if aircraft address has the value 2 and ground-vector-modulus has the value 3, then the aircraft address element will be included in the 1st, 3rd, 5th, 7th etc. ADS-reports, and the ground-vector will be included in the 1st, 4th, 7th, 10th etc. ADS-reports.

Note 2.— Position, time-stamp and fom will always be included in every ADS-report.

2.2.1.7.4.4.6 When short-term-intent is included in the *report details* parameter of an ADS-report, the ADS-air-user shall insert a value that covers way points and estimated times of arrival for the following *intent-projection-time* as measured from the timestamp on the ADS-report.

2.2.1.7.4.4.7 When extended-projected-profile is included in the *report details* parameter of an ADS-report, the ADS-air-user shall insert a value that contains extended projected profile information for the *time-interval* or the *number-of-way-points* indicated in the *extended-projected-profile-modulus*.

2.2.1.7.4.4.8 When the ADS-air-user invokes ADS-report request in response to an ADS-periodic-contract indication, the *contract type* parameter shall be set to *periodic-contract*.

Note 1.— Apart from circumstances detailed in 2.2.1.7.4.2.1, the positive acknowledgement parameter is not present in the ADS-report request.

Note 2.— When the ADS-air-user invokes ADS-report request in response to an ADS-periodic-contract indication, the event type parameter is not be included in the ADS-report request.

2.2.1.7.5 Ground Cancellation of Contracts

Note 1.— 2.2.1.7.5 details the actions taken by the ADS-ground-user and the ADS-air-user in the cancellation of contracts.

Note 2.— When an ADS-ground-user requires to cancel an event contract or a periodic contract, then it either invokes ADS-cancel request with the contract type parameter set to event-contract or periodic-contract respectively. When an ADS-ground-user requires to cancel all contracts with the aircraft it invokes ADS-cancel-all-contracts request

2.2.1.7.5.1 If the ADS-air-user receives an ADS-cancel-contract with *contract type* parameter set to *event-contract*, the ADS-air-user shall cancel any event contract with that ground system.

2.2.1.7.5.2 If the ADS-air-user receives an ADS-cancel-contract with *contract type* parameter set to *periodic-contract*, the ADS-air-user shall cancel any periodic contract with that ground system.

2.2.1.7.5.3 When the ADS-air-user receives an ADS-cancel-all-contracts indication, it shall cancel all contracts (event, periodic and emergency) with that ground system.

Note.— There is no provision for cancellation of demand contracts.

2.2.1.7.6 Establishment and Operation of Emergency Contracts

Note 1.— 2.2.1.7.6 details the actions taken by the ADS-ground-user and the ADS-air-user in the establishment and operation of emergency contracts.

Note 2.— The emergency contract is only air user activated, and may be initiated either by human or automatically by the aircraft system.

2.2.1.7.6.1 On emergency contract initiation, the ADS-air-user shall establish an emergency contract with every ground system with which it has an event contract or a periodic contract (or both).

2.2.1.7.6.2 When an ADS-periodic-contract indication or ADS-event-contract indication occurs during an emergency, from an ADS-ground-user with which the aircraft has not got an event contract or a periodic contract, then the ADS-air-user shall:

a) acknowledge the contract in the manner indicated in 2.2.1.7.3 and 2.2.1.7.4, with either a response or an ADS-report request, and

b) if a negative acknowledgement is not sent, establish an emergency contract with the ADS-ground-user.

2.2.1.7.6.3 When the ADS-air-user establishes an emergency contract with an ADS-ground-user, then:

2.2.1.7.6.3.1 If the ADS-air-user has a periodic contract with the ADS-ground-user, then the ADS-air-user shall suspend the operation of the periodic contract.

2.2.1.7.6.3.2 If the ADS-air-user has no periodic contract with the ADS-ground-user at the time of establishing the emergency contract, then the ADS-air-user shall set the emergency reporting rate to be 60 seconds.

2.2.1.7.6.3.3 If the ADS-air-user has a periodic contract with the ADS-ground-user at the time of establishing the emergency contract, then the ADS-air-user shall set the emergency reporting rate to be as indicated in Table 2.2.1.7-2.

Table 2.2.1.7-2: Emergency reporting rate calculation when a periodic contract is in place

Existing periodic reporting rate	Emergency reporting rate
1 second	1 second
greater than two minutes	60 seconds

Existing periodic reporting rate	Emergency reporting rate
less than or equal to two minutes and greater than 1 second	half the reporting rate of the periodic contract rounded down to the nearest second

2.2.1.7.6.3.4 The ADS-air user shall invoke ADS-emergency-report request at the emergency reporting rate.

2.2.1.7.6.3.5 Subject to the restrictions stated in 2.2.1.7.1.5.5, the ADS-air user shall include the following elements in the emergency report details parameter in the ADS-emergency-report request:

a) *position*, *timestamp* and *fom*; and

b) *ground-vector* and *aircraft address* for the first ADS-emergency-report request after the emergency contract has been established, and subsequently every fifth ADS-emergency-report request.

Note 1.— That is, on the 1st, 6th, 11th, 16th etc. ADS-emergency-report request.

Note 2.— Apart from conditions specified in 2.2.1.7.7, the positive acknowledgement parameter is not present in any ADS-emergency-report request.

2.2.1.7.6.3.6 If the ADS-air-user receives an ADS-periodic-contract indication from an ADS-ground-user with which the ADS-air-user has an emergency contract, then the ADS-air-user shall invoke ADS-periodic-contract response giving a reply that would be appropriate if the emergency contract were not in operation.

Note.— This implies that a reply of an ADS-report is not possible.

2.2.1.7.6.3.7 For each ground system, the ADS-air-user shall store details of the most recent of the following:

a) ADS-periodic-contract indication, which had a positive acknowledgement as a response;

b) ADS-periodic-contract indication, which had a noncompliance notification as a response; and

c) ADS-cancel-contract indication with value *periodic-contract*.

Note.— This information is used to re-establish the periodic contract after the emergency is over.

2.2.1.7.7 Modifying an Emergency Contract

Note 1.— 2.2.1.7.7 details the actions taken by the ADS-ground-user and the ADS-air-user when modifying an emergency contract.

Note 2.— When the ADS-ground-user requires to modify the reporting rate of an emergency contract it invokes ADS-modify-emergency-contract request.

2.2.1.7.7.1 When the ADS-air-user receives an ADS-modify-emergency-contract indication, and it is able to comply with the request, it shall:

a) change the emergency reporting rate to the time indicated in the reporting interval parameter; and

b) include a *positive acknowledgement* parameter in the next ADS-emergency-report request.

2.2.1.7.7.1.1 **Recommendation.—** *the ADS-air-user should invoke the next ADS-emergency-report request within 0.5 seconds.*

2.2.1.7.7.2 When the ADS-air-user receives an ADS-modify-emergency-contract indication, and it is not able comply with the request, then the ADS-air-user shall invoke ADS-modify-emergency-contract response.

2.2.1.7.7.2.1 **Recommendation.—** *the ADS-air-user should invoke the next ADS-emergency-report request within 0.5 second.*

Note 1.— The emergency reporting rate remains unchanged.

Note 2.— The existing five ADS-emergency-report cycle remains regardless of any reporting rate modification, moreover, the position within the cycle also remains unaffected. For example, if the second ADS-emergency-report request was invoked before the modification of emergency reporting rate, the following ADS-emergency-report request will be the third.

2.2.1.7.8 Cancellation of an Emergency Contract

Note 1.— 2.2.1.7.8 details the actions taken by the ADS-ground-user and the ADS-air-user when the ADS-air-user cancels emergency contracts.

Note 2.— The initiation of the cancellation of an emergency contract may only be done by human intervention in the aircraft.

2.2.1.7.8.1 When the ADS-air-user cancels emergency contracts, it shall cancel the emergency contract with each ADS-ground-user with which it has an emergency contract.

2.2.1.7.8.2 When the ADS-air-user cancels an emergency contract, it shall:

a) invoke ADS-cancel-emergency request,

b) if a periodic contract was in operation before the emergency contract was established and no ADS-periodic-contract indications were accepted and no

ADS-cancel-contract indications (with a value of periodic contract) were received during the emergency contract, then resume operation of the periodic contract, and

c) if the latest event to be stored (as indicated in 2.2.1.7.6.3) was an ADS-periodic-contract indication, then initiate the operation of that periodic contract.

Note.— If the latest event to be stored (as indicated in 2.2.1.7.6.3) was an ADS-cancel-contract indication (with a value of periodic-contract) then no periodic contract is started or resumed.

2.2.1.7.8.2.1 **Recommendation.**— *If the ADS-air-user reinstates a periodic contract or initiates a new periodic contract, it should invoke the next ADS-report request within 0.5 second.*

2.2.1.7.8.2.2 If the ADS-air-user reinstates a periodic contract, it shall restart in the same position in the cycle of reports as it was in when the emergency contract was established.

2.2.1.7.9 Operation of Aborts

Note 1.— 2.2.1.7.9 details the actions taken by an ADS-ground-user and the ADS-air-user aborts occur.

Note 2.— When an ADS-ground-user or and ADS-air-user requires to abort the current contracts, it initiates ADS-user-abort request.

2.2.1.7.9.1 When the ADS-air-user or the ADS-ground-user receives an ADS-user-abort indication or an ADS-provider-abort-indication, it shall cancel all contracts with the peer ADS-user.

2.2.1.7.10 Parameter Value Unit, Range and Resolution

2.2.1.7.10.1 An ADS user shall interpret ADS parameter value unit, range and resolution as defined in 2.2.1.4.

2.2.1.8 Subsetting Rules

2.2.1.8.1 General

Note.— 2.2.1.8 specifies conformance requirements which all implementations of the ADS protocol obey.

2.2.1.8.1.1 An implementation of either the ADS ground based service or the ADS air based service claiming conformance to 2.2.1 shall support the ADS protocol features as shown in the tables below.

Note.— The 'status' column indicates the level of support required for conformance to the ADS-ASE protocol described in 2.2.1. The values are as follows:

a) ***'M'*** *mandatory support is required;*

b) ***'O'*** *optional support is permitted for conformance to the ADS protocol;*

c) ***'N/A'*** *the item is not applicable; and*

d) ***'C.n'*** *the item is conditional where n is the number which identifies the condition which is applicable.*

Table 2.2.1.8-1. ADS Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 2.2.1.8-2. ADS Protocol Options

	Status	**Associated Predicate**
ADS-air-ASE	C.1	ADS/air
ADS-ground-ASE	C.1	ADS/ground
Demand contract supported	if (ADS/air) M, else C.2	DC-FU
Event contract supported	if (ADS/air) M, else C.2	EC-FU
Periodic contract supported	if (ADS/air) M, else C.2	PC-FU
Emergency contract supported	if (EC-FU or PC-FU) M, else N/A	EM-FU

C.1: a conforming implementation shall support one and only one of these two options.
C.2: optional; a conforming implementation shall support at least one of the three options.

Table 2.2.1.8-3. ADS-ground-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I	DC-FU + ADS/ground	ADS-ground-ASE supporting demand contract only
II	EC-FU + EM-FU+ ADS/ground	ADS-ground-ASE supporting event and emergency contracts
III	PC-FU + EM-FU+ ADS/ground	ADS-ground-ASE supporting periodic and emergency contracts
IV	DC-FU + EC-FU + EM-FU+ ADS/ground	ADS-ground-ASE supporting demand, event and emergency contracts
V	DC-FU + PC-FU + EM-FU+ ADS/ground	ADS-ground-ASE supporting demand, periodic and emergency contracts

	List of Predicates	**Functionality Description**
VI	EC-FU + PC-FU + EM-FU+ ADS/ground	ADS-ground-ASE supporting event, periodic and emergency contracts
VII	DC-FU + EC-FU + PC-FU+ EM-FU + ADS/ground	ADS-ground-ASE supporting demand, event, periodic and emergency contracts

Table 2.2.1.8-4. ADS-air-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I	DC-FU + EC-FU + PC-FU+ EM-FU + ADS/air	ADS-air-ASE supporting demand, event, periodic and emergency contracts

Table 2.2.1.8-5. Supported ADS Service Primitives

	Sending (req,[cnf])	**Receiving (ind, [rsp])**
ADS-demand-contract	if (ADS/ground and DC-FU) M, else N/A	if (ADS/air) M, else N/A
ADS-event-contract	if (ADS/ground and EC-FU) M, else N/A	if (ADS/air) M, else N/A
ADS-periodic-contract	if (ADS/ground and PC-FU) M, else N/A	if (ADS/air) M, else N/A
ADS-report	if (ADS/air) M, else N/A	if (ADS/ground) M, else N/A
ADS-cancel	if (ADS/ground and (EC-FU or PC-FU)) M, else N/A	if (ADS/air) M, else N/A
ADS-cancel-all-contracts	if (ADS/ground and (EC-FU or PC-FU)) M, else N/A	if (ADS/air) M, else N/A
ADS-emergency-report	if (ADS/air) M, else N/A	if (ADS/ground and EM-FU) M, else N/A
ADS-modify-emergency-contract	if (ADS/ground and EM-FU) M, else N/A	if (ADS/air) M, else N/A
ADS-cancel-emergency-contract	if (ADS/air) M, else N/A	if (ADS/ground and EM-FU) M, else N/A

	Sending (req,[cnf])	**Receiving (ind, [rsp])**
ADS-user-abort	M	M
ADS-provider-abort	N/A	M

Table 2.2.1.8-6. Supported ADS APDUs

	Sender	**Receiver**
[ADSAircraftPDUs] aDS-cancel-emergency-PDU	if (ADS/air) M, else N/A	if (ADS/ground and EM-FU) M, else N/A
[ADSAircraftPDUs] aDS-demand-report-PDU	if (ADS/air) M, else N/A	if (ADS/ground and DC-FU) M, else N/A
[ADSAircraftPDUs] aDS-emergency-report-PDU	if (ADS/air) M, else N/A	if (ADS/ground and EM-FU) M, else N/A
[ADSAircraftPDUs] aDS-event-report-PDU	if (ADS/air) M, else N/A	if (ADS/ground and EC-FU) M, else N/A
[ADSAircraftPDUs] aDS-negative-acknowledgement -PDU	if (ADS/air) M, else N/A	if (ADS/ground) M, else N/A
[ADSAircraftPDUs] aDS-noncompliance-notification -PDU	if (ADS/air) M, else N/A	if (ADS/ground) M, else N/A
[ADSAircraftPDUs] aDS-periodic-report-PDU	if (ADS/air) M, else N/A	if (ADS/ground and PC-FU) M, else N/A
[ADSAircraftPDUs] aDS-positive-acknowledgement-PDU	if (ADS/air) M, else N/A	if (ADS/ground and EM-FU) M, else N/A
[ADSAircraftPDUs] aDS-provider-abort-PDU	if (ADS/air) M, else N/A	if (ADS/ground) M, else N/A
[ADSGroundPDUs] aDS-cancel-all-contracts-PDU	if (ADS/ground and EM-FU) M, else N/A	if (ADS/air) M, else N/A
[ADSGroundPDUs] aDS-cancel-contract-PDU	if (ADS/ground and EM-FU) M, else N/A	if (ADS/air) M, else N/A

	Sender	**Receiver**
[ADSGroundPDUs] aDS-cancel-emergency-acknowledgement-PDU	if (ADS/ground and EM-FU) M, else N/A	if (ADS/air) M, else N/A
[ADSGroundPDUs] aDS-demand-contract-PDU	if (ADS/ground and DC-FU) M, else N/A	if (ADS/air) M, else N/A
[ADSGroundPDUs] aDS-event-contract-PDU	if (ADS/ground and EC-FU) M, else N/A	if (ADS/air) M, else N/A
[ADSGroundPDUs] aDS-modify-emergency-contract-PDU	if (ADS/ground and EM-FU) M, else N/A	if (ADS/air) M, else N/A
[ADSGroundPDUs] aDS-periodic-contract-PDU	if (ADS/ground and PC-FU) M, else N/A	if (ADS/air) M, else N/A
[ADSGroundPDUs] aDS-provider-abort-PDU	if (ADS/ground) M, else N/A	if (ADS/air) M, else N/A

2.2.2 AUTOMATIC DEPENDENT SURVEILLANCE REPORT FORWARDING APPLICATION

2.2.2.1 Introduction

2.2.2.1.1 The ADS report forwarding application will allow users to obtain positional and other information from suitably equipped aircraft in a timely manner in accordance with their requirements

Note 1.— Structure of 2.2.2

2.2.2 defines the ground-ground aspects of ADS only. 2.2.1 defines the air-ground communication aspects of ADS:

a) 2.2.2.1: INTRODUCTION contains the 2.2.2's purpose, structure, and a summary of the functions of ADS.

b) 2.2.2.2: GENERAL REQUIREMENTS contains backwards compatibility and error processing requirements.

c) 2.2.2.3: THE ABSTRACT SERVICE contains the description of the abstract service provided by the application service elements (ASE) defined for ADS Report Forwarding.

d) 2.2.2.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by ADS-RF-ASEs using Abstract Syntax Notation Number One (ASN.1).

e) 2.2.2.5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the ADS protocol, as well as time constraints and the exception handling procedures associated with these exchanges. 2.2.2.5 describes also the ADS protocol related to report forwarding in terms of state tables.

f) 2.2.2.6: COMMUNICATION REQUIREMENTS contains the requirements that the ADS-RF-ASEs imposes on the underlying communication system.

g) 2.2.2.7: ADS USER REQUIREMENTS outlines the requirements that a user of an ADS-RF-ASE must meet.

h) 2.2.2.8: SUBSETTING RULES provides rules for subsetting the ADS Report Forwarding SARPs.

Note 2.— General Functionality

a) It will be necessary for an implementation to provide information which is both accurate and timely in the ADS reports; however, quantification of the age and accuracy of the information is beyond the scope of 2.

Note 3.— Establishment and Operation of Forward Contract

a) Functional Description

1) This function provides a method for a ground system to establish a forward contract with another ground system and to forward ADS reports. This function is initiated by a ground system having received ADS reports, which then forwards the received ADS reports to another ground system.

2) The receiving ground system may reject the ADS start forwarding request.

3) When an ADS report is to be sent the ground system will use an ADS forward report message.

b) Message Descriptions

1) The ADS start forwarding request message may contain the first ADS forward report.

2) The ADS start forwarding response contains the result of the establishment of the forward contract.

3) An ADS forward report message contains the aircraft address and flight identification of the aircraft the report is related to, and either a periodic, event, demand, or emergency report.

Note 4.— Cancellation of the Forward Contract

a) Functional Description

1) This function allows the sending ground system to cancel the Forward Contract.

2) The sending ground system sends a cancel forward contract message to the receiving ground system.

b) Message Descriptions

1) The cancel forward contract message does not contain any information.

2.2.2.2 General Requirements

2.2.2.2.1 ADS-RF-ASE Version Number

2.2.2.2.1.1 The ADS-RF-ASE version number shall be set to one.

2.2.2.2.2 Error Processing Requirements

2.2.2.2.2.1 In the event of information input by the ARF-user being incompatible with that able to be processed by the system, the ARF-user shall be notified.

2.2.2.2.2.2 In the event of a ARF-user invoking an ADS Report Forwarding service primitive, when the ADS-RF-ASE is not in a state specified in 2.2.2.5, the ARF-user shall be notified.

2.2.2.3 The Abstract Service

2.2.2.3.1 Service Description

2.2.2.3.1.1 An implementation of the ADS Report Forwarding service shall exhibit external behaviour consistent with having implemented an ADS-RF-ASE.

Note 1.— 2.2.2.3 defines the abstract service interface for the ADS Report Forwarding service. The ADS-RF-ASE abstract service is described in 2.2.2.3 from the viewpoint of the ADS-RF-user and the ADS-RF service-provider.

Note 2.— 2.2.2.3 defines the static behaviour (i.e. the format) of the ADS Report Forwarding abstract service. Its dynamic behaviour (i.e. how it is used) is described in 2.2.2.7.

Note 3.— Figure 2.2.2.3-1 shows the functional model of the ADS Report Forwarding Function. The functional modules identified in this model are the following :

a) the ADS Report Forwarding user,

b) the ADS Report Forwarding Application Entity (ADS-RF-AE) service interface,

c) the ADS-RF-AE,

d) the ADS Report Forwarding Control Function (ADS-RF-CF),

e) the ADS Report Forwarding Application Service Element (ADS-RF-ASE) service interface,

f) the ADS-RF-ASE, and

g) the Dialogue Service (DS) interface.

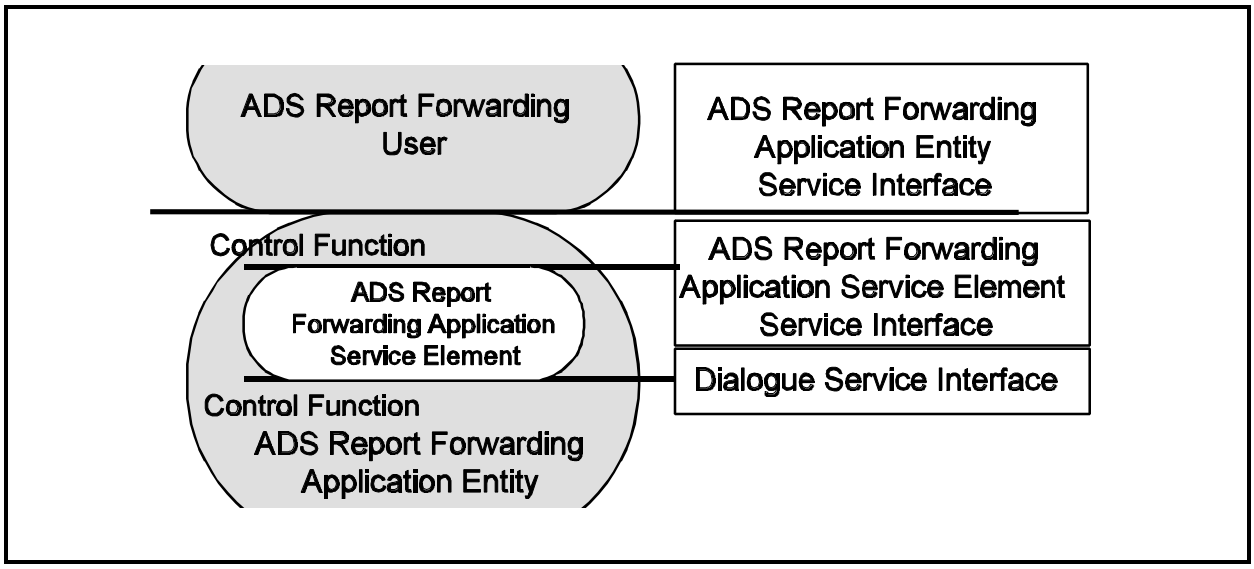

Figure 2.2.2.3-1. Functional Model of the ADS Report Forwarding Function

Note 4.— The ADS-RF-user represents the operational part of the ADS system. This user does not perform the communication functions but relies on a communication service provided to it via the ADS-RF-AE through the ADS-RF-AE service interface. The individual actions at this interface are called ADS-RF-AE service primitives. Similarly, individual actions at other interfaces in the communication system are called service primitives at these interfaces.

Note 5.— The ADS-RF-AE consists of several elements, including the ADS-RF-ASE and the ADS-RF-CF. The DS interface is made available by the ADS-RF-CF to the ADS-RF-ASE for communication with the peer ADS-RF-ASE.

Note 6.— The ADS-RF-ASE is the element in the communication system which executes the ADS-RF specific protocol. In other words, it takes care of the ADS-RF specific service primitive sequencing actions, message creation, timer management, error and exception handling.

Note 7.— The ADS-RF-ASE interfaces only with the ADS-RF-CF. This ADS-RF-CF is responsible for mapping service primitives received from one element (such as the ADS-RF-ASE and the ADS-RF-user) to other elements which interface with it. The part of the ADS-RF-CF which is relevant from the point of view of these SARPs, i.e. the part between the ADS-RF-user and the ADS-RF-ASE, will map ADS-RF-AE service primitives to ADS-RF-ASE service primitives transparently.

Note 8.— The DS interface is the interface between the ADS-RF-ASE and part of ADS-RF-CF underneath, the ADS-RF-ASE and provides the dialogue service.

2.2.2.3.2 The ADS-RF-ASE Abstract Service

Note.— There is no requirement to implement the service in an ADS product; however, it is necessary to implement the ground based and air based system in such a way that it will be impossible to detect (from the peer system) whether or not an interface has been built.

2.2.2.3.2.1 The ADS-RF-ASE abstract service shall consist of a set of the following services as allowed by the subsetting rules defined in 2.2.2.8:

a) ADS-start-forward service as defined in 2.2.2.3.4;

b) ADS-forward-report service as defined in 2.2.2.3.5;

c) ADS-end-forward service as defined in 2.2.2.3.6;

d) ADS- user-abort service as defined in 2.2.2.3.7;

e) ADS-provider-abort service as defined in 2.2.2.3.8.

Note.— An abstract syntax is a syntactical description of a parameter which does not imply a specific implementation. Only when the ADS-RF-ASE maps a parameter onto an APDU field, or vice-versa, is the abstract syntax of the parameter described by using the ASN.1 of 2.2.2.4 for this field.

2.2.2.3.3 Conventions

Note 1.— For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables 2.2.2.3:

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.

Note 2.— The following abbreviations are used in this 2.2.2.3:

a) Req request; data is input by an ADS-RF-user initiating the service to its respective ASE;

b) Ind indication; data is indicated by the receiving ASE to its respective ADS-RF-user;

c) Rsp response; data is input by receiving ADS-RF-user to its respective ASE;

d) Cnf confirmation; data is confirmed by the initiating ASE to its respective ADS-RF-user.

Note 3.— An unconfirmed service allows just one message to be transmitted, in one direction,.

Note 4.— A confirmed service provides end-to-end confirmation that a message sent by one user was received by its peer user.

2.2.2.3.4 ADS-start-forward Service

Note.— The ADS-start-forward service allows an ADS-RF-user to request the establishment of a forward contract with another ADS-RF-user. An ADS report may be included within this service. It is a confirmed service, initiated by an ADS-RF-user.

2.2.2.3.4.1 The ADS-start-forward service shall contain primitives and parameters as presented in Table 2.2.2.3-1 where the version numbers of the peer ASEs are compatible.

2.2.2.3.4.2 The ADS-start-forward service shall contain primitives and parameters as presented in Table 2.2.2.3-2 where the version numbers of the peer ASEs are incompatible.

Table 2.2.2.3-1. ADS-start-forward service parameters - compatible version numbers

Parameter Name	Req	Ind	Cnf
ICAO Facility designation	M		
Class of communication service	U		
Forwarded report details	U	C(=)	
Reply			M

Table 2.2.2.3-2. ADS-start-forward service parameters - incompatible version numbers

Parameter Name	Req	Cnf
ICAO Facility designation	M	
Class of communication service	U	
Forwarded report details	U	
Reply		M

2.2.2.3.4.3 ICAO Facility designation

Note.— This parameter contains the receiving ground system's ICAO facility designation.

2.2.2.3.4.3.1 The *ICAO Facility designation* parameter value shall conform to the abstract syntax four- to eight-character ICAO facility designation.

2.2.2.3.4.4 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service, if specified by the ADS-RF-user.

2.2.2.3.4.4.1 Where specified by the ADS-RF-user, the *class of communication service* parameter shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G” or “H”.

Note.— Where not specified by the ADS-RF-user, this indicates that there will be no routing preference.

2.2.2.3.4.5 Forwarded Report Details

Note.— This parameter contains the details of the forwarded ADS report.

2.2.2.3.4.5.1 The *forwarded report details* parameter value shall conform to the ASN.1 abstract syntax *ADSForwardedReport*.

2.2.2.3.4.6 Reply

Note.— This parameter indicates whether the ADS-start-forward request has been accepted (abstract value is “accepted”) or rejected (abstract value is “incompatible version”) by the peer ADS-RF-user.

2.2.2.3.4.6.1 The *Reply* parameter value shall have one of the following abstract values:

a) “accepted”, or

b) “incompatible version”.

2.2.2.3.5 ADS-forward-report Service

Note.— The ADS-forward-report service allows an ADS-RF-user to forward an ADS report to another ADS-RF-user. This is an unconfirmed service, initiated by the ADS-RF-user which has initiated the ADS-start-forward service.

2.2.2.3.5.1 The ADS-forward-report service shall contain primitives and parameters as contained in Table 2.2.2.3-3.

Table 2.2.2.3-3. ADS-forward-report service parameters

Parameter Name	Req	Ind
Forwarded Report details	M	M(=)

2.2.2.3.5.2 Forwarded Report Details

Note.— This parameter contains the details of the forwarded ADS report.

2.2.2.3.5.2.1 The *forwarded report details* parameter value shall conform to the ASN.1 abstract syntax *ADSForwardedReport*.

2.2.2.3.6 ADS-end-forward Service

Note.— The ADS-end-forward service allows the ADS-RF-user forwarding the ADS reports to end the ADS Report Forwarding service. It is a unconfirmed service, initiated by the sending ADS-RF-user.

2.2.2.3.6.1 The ADS-end-forward service shall contain primitives as contained in Table 2.2.2.3-4.

Table 2.2.2.3-4. ADS-end-forward service parameters

Parameter Name	Req	Ind
none		

2.2.2.3.7 ADS-user-abort Service

Note 1.— The ADS-user-abort service allows the ADS-RF-user to abort a forward contract. It is an unconfirmed service, initiated by an ADS-RF-user. Messages in transit may be lost during this operation. It can be invoked at any time that the ADS-RF-user is aware that any ADS Report Forwarding service is in operation.

Note 2.— If the service is invoked prior to complete establishment of the dialogue, the ADS-user-abort indication may not be provided. An ADS-provider-abort indication may result instead.

2.2.2.3.7.1 The ADS-user-abort service shall contain primitives as contained in Table 2.2.2.3-5.

Table 2.2.2.3-5. ADS-user-abort service parameters

Parameter Name	Req	Ind
none		

2.2.2.3.8 ADS-provider-abort Service

Note.— The ADS-provider-abort service allows the ADS-service-provider to inform the ADS-RF-users that it can no longer provide the ADS Report Forwarding service for a particular ADS-RF-user pairing. It is initiated by the ADS-service-provider. Messages in transit may be lost during this operation.

2.2.2.3.8.1 The ADS-provider-abort service shall contain primitives and parameters as contained in Table 2.2.2.3-6.

Table 2.2.2.3-6. ADS-provider-abort service parameters

Parameter Name	Ind
Reason	M

2.2.2.3.8.2 Reason

This parameter identifies the reason for the abort.

2.2.2.3.8.2.1 The *reason* parameter shall conform to the ASN.1 abstract syntax *AbortReason*.

2.2.2.4 Formal Definitions of Messages

2.2.2.4.1 Encoding/Decoding Rules

2.2.2.4.1.1 An ADS-RF-ASE shall be capable of encoding and decoding [ADSRFPDUs] APDUs.

2.2.2.4.2 ADS ASN.1 Abstract Syntax

2.2.2.4.2.1 The abstract syntax of the ADS-RF protocol data units shall comply with the description contained in the ASN.1 module ADSRFMessageSetVersion1 (conforming to ISO/IEC 8824), as defined in 2.2.2.4.

```
ADSRFMessageSetVersion1 DEFINITIONS ::=

BEGIN

IMPORTS
        AbortReason, ADSEmergencyReport, ADSReport,AircraftAddress, EventTypeReported
                        FROM ADSMessageSetVersion1;
ADSRFPDUs ::= CHOICE
{
        aDS-forwarded-report-PDU        [0]     ADSForwardedReport,
        aDS-provider-abort-PDU          [1]     AbortReason,
...
}
ADSForwardedReport ::= SEQUENCE
{
        aircraftAddress                         AircraftAddress,
        forwardedADSReport                      ForwardedReport
}
ForwardedReport::= CHOICE
{
        aDSDemandReport                 [0]     ADSReport,
        aDSPeriodicReport               [1]     ADSReport,
        aDSEventReport                  [2]     SEQUENCE
                                                {
                                                        event-type      EventTypeReported,
                                                        aDSReport       ADSReport
                                                },
aDSEmergencyReport                      [3]     ADSEmergencyReport
}
END -- of ADSRFMessageSetVersion1
```

2.2.2.5 Protocol Definition

2.2.2.5.1 Sequence Rules

2.2.2.5.1.1 Only the sequence of primitives defined illustrated in figures 2.2.2.5-1 to 2.2.2.5-6 shall be permitted.

Note 1.— The following figures define the valid sequences of primitives that are possible to be invoked during the operation of the ADS Report Forwarding function. 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.— Abort primitive may interrupt and terminate any of the normal message sequences outlined below.

Note 3.— Primitives are processed in the order in which they are received (see 4.3.3.1.2.4).

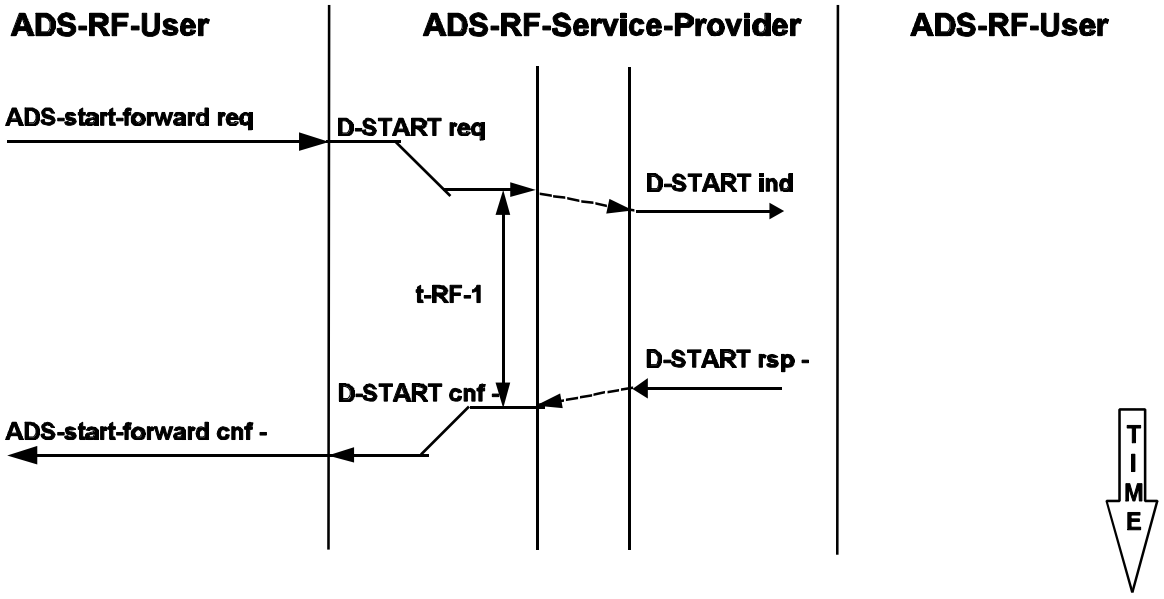

Figure 2.2.2.5-1. Use of forward contract with negative response

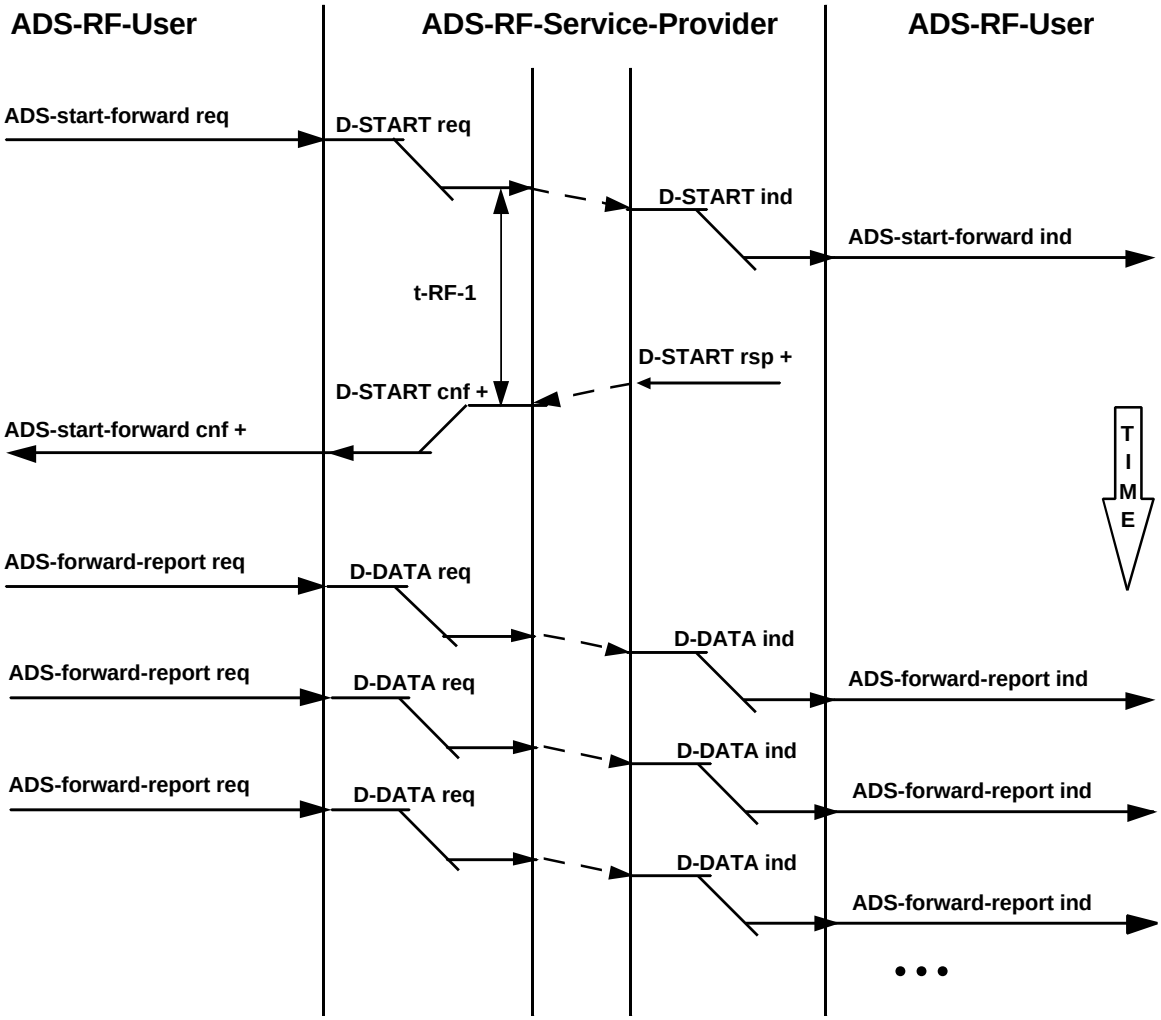

Figure 2.2.2.5-2. Use of forward contract with positive response

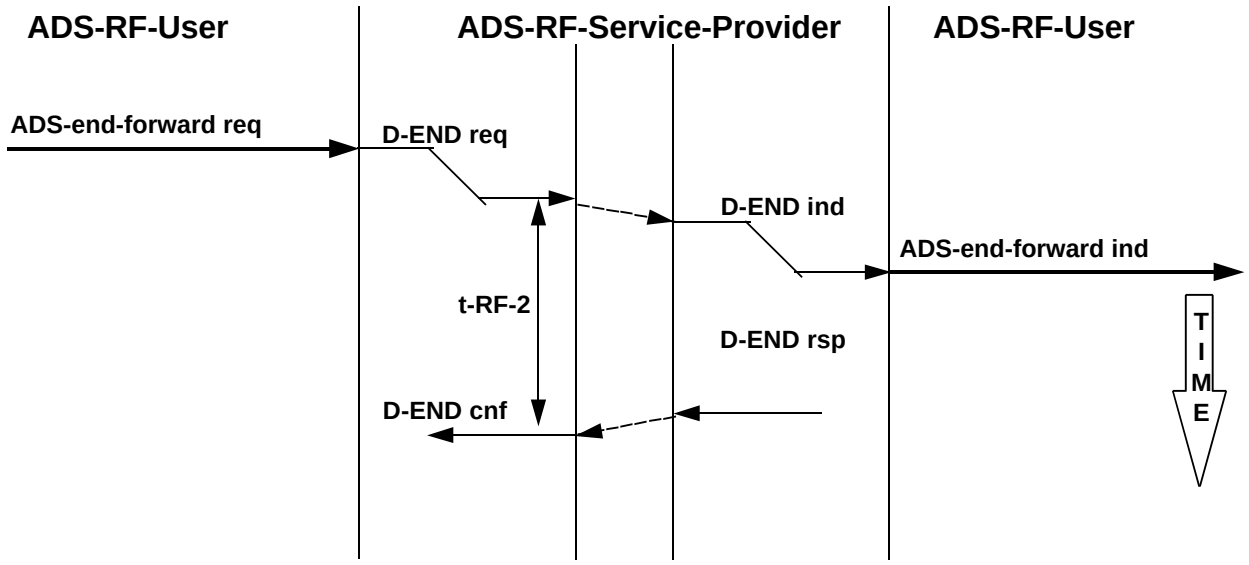

Figure 2.2.2.5-3. Use of end forward service

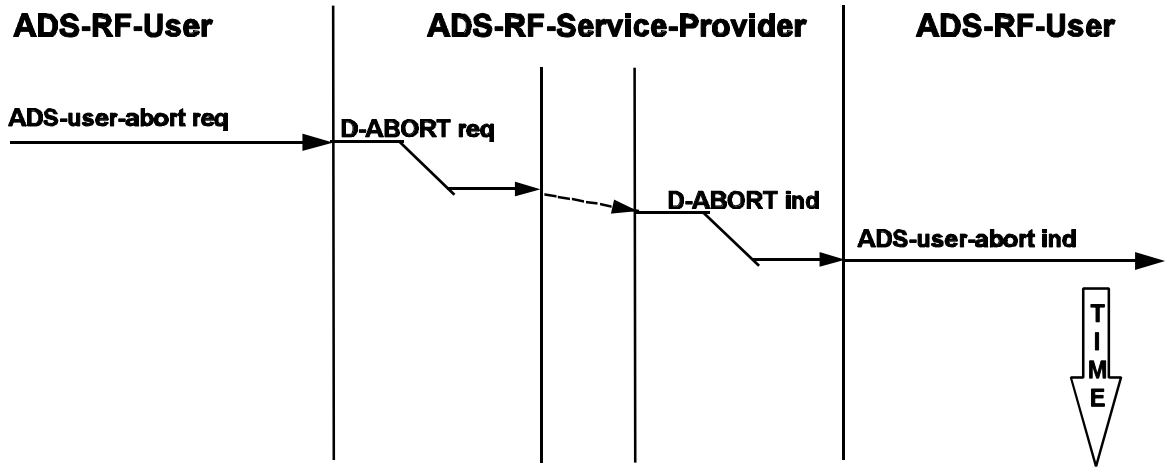

Figure 2.2.2.5-4. ADS-RF-user abort service with a Forward contract in place

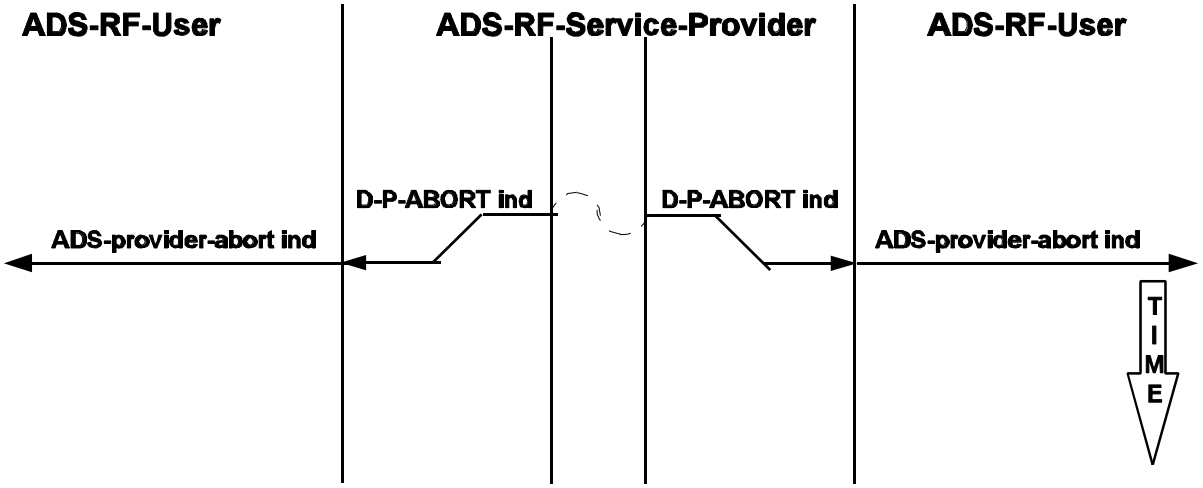

Figure 2.2.2.5-5. Dialogue service provider abort service, with forward contract in place

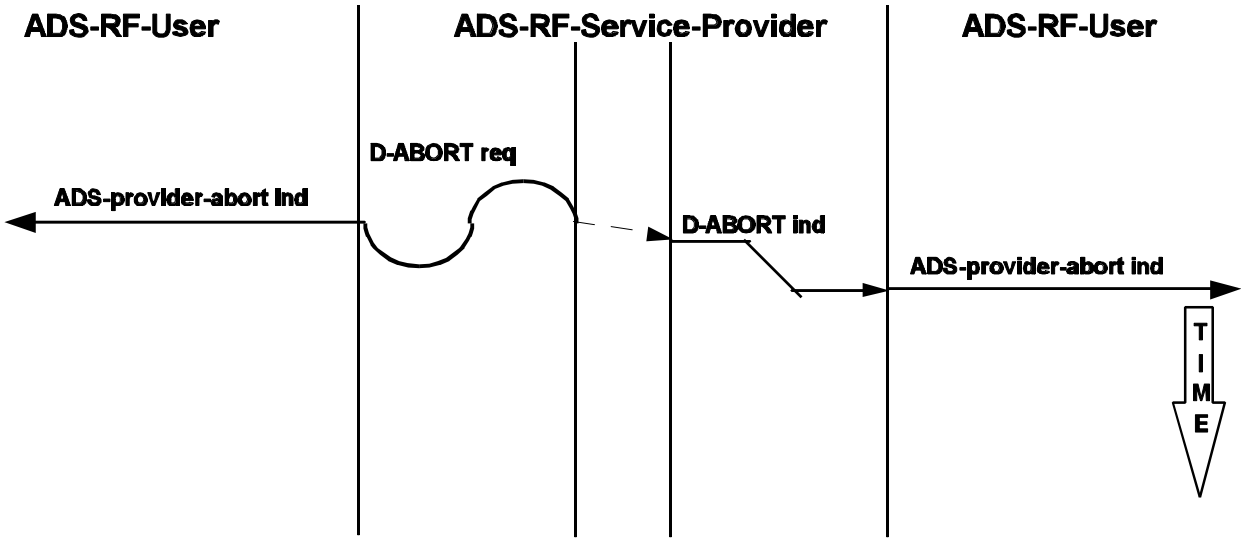

Figure 2.2.2.5-6. ADS-RF-ASE abort with forward contract in place

2.2.2.5.2 ADS RF Service Provider Timers

2.2.2.5.2.1 The ADS-ASE shall be capable of detecting when a timer expires.

Note 1.— Table 2.2.2.5-1 lists the time constraints related to the ADS Report Forwarding function. Each time constraint requires a timer to be set in the ADS protocol machine.

Note 2.— If the timer expires before the final event has occurred, the ADS-RF-ASE takes the appropriate action.

2.2.2.5.2.2 **Recommendation**. — *The timer values should be as indicated in Table 2.2.2.5-1.*

Table 2.2.2.5-1: ADS RF Service Provider Timers

ADS Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
ADS forward contract	t-RF-1	6 minutes	ADS-start-forward request	ADS-start-forward confirmation
	t-RF-2	6 minutes	D-END request	D-END confirmation

2.2.2.5.3 ADS-ASE Protocol Description

Note.— 2.2.2.5.3 defines the protocol for the ADS-RF-ASE. The protocol for the initiating ADS-RF-ASE and the responding ADS-RF-ASE are given separately.

2.2.2.5.3.1 If an APDU is not received when one is required, or one is received in an inappropriate dialogue service primitive, then the exception handling procedures as described in 2.2.2.5.4.3 shall apply.

2.2.2.5.3.2 Upon receipt of an APDU, if no actions are described for the arrival of that APDU when in a particular state, then the exception handling procedures as described in 2.2.2.5.4.4 shall apply.

2.2.2.5.3.3 Upon receipt of an APDU that cannot be decoded, then the exception handling procedures as described in 2.2.2.5.4.6 shall apply.

2.2.2.5.3.4 ADS Initiating RF ASE

Note.— The initiating ADS-RF-ASE has the following states:

a) RF-I-IDLE

b) RF-I-START

c) RF-I-ACTIVE

d) RF-I-END

2.2.2.5.3.4.1 On initiation, the initiating ADS-RF-ASE shall be in the RF-I-IDLE state.

2.2.2.5.3.4.2 Upon receipt of an ADS-start-forward request:

2.2.2.5.3.4.2.1 If in the RF-I-IDLE state, the ADS-RF-ASE shall:

a) invoke D-START request with parameters as defined in Table 2.2.2.5-2;

b) start the t-RF-1 timer, and

c) enter the RF-I-START state.

Table 2.2.2.5-2. D-START request parameter values

Parameter name	Derivation of Parameter Value
Called peer Id	*ICAO Facility designation* parameter value from ADS-forward-contract request
Calling peer Id	Not used
DS-user version number	Not used
Security requirements	Not used
Quality of service	Routing class: ATSC, with value from *Class of communication service* parameter value from ADS-forward-contract request Priority: High priority flight safety messages RER: Low
User data	The *Forwarded report details* parameter value, if provided.

2.2.2.5.3.4.3 Upon receipt of an ADS-forward-report request:

2.2.2.5.3.4.3.1 If in the RF-I-ACTIVE state, the initiating ADS-RF-ASE shall:

a) invoke D-DATA request with parameters as defined in Table 2.2.2.5-3, and

b) remain in the RF-I-ACTIVE state.

Table 2.2.2.5-3

Parameter Name	Derivation of Parameter Value
User data	The *Forwarded report details* parameter value

2.2.2.5.3.4.4 Upon receipt of an ADS-end-forward request:

2.2.2.5.3.4.4.1 If in the RF-I-ACTIVE state, the initiating ADS-RF-ASE shall:

a) invoke D-END request with parameters as defined in Table 2.2.2.5-4,

b) start the t-RF-2 timer, and

c) enter the RF-I-END state.

Table 2.2.2.5-4

Parameter Name	Derivation of Parameter Value
User data	Not provided

2.2.2.5.3.4.5 Upon receipt of an ADS-user-abort request:

2.2.2.5.3.4.5.1 If not in the RF-I-IDLE state, the initiating ADS-RF-ASE shall:

a) stop any timers,

b) invoke D-ABORT request with parameters as defined in Table 2.2.2.5-5,

c) enter the RF-I-IDLE state.

Table 2.2.2.5-5

Parameter Name	Derivation of Parameter Value
Originator	“user”
User data	Not provided

2.2.2.5.3.4.6 Upon receipt of a D-START confirmation with a *Result* parameter value of “accepted”:

2.2.2.5.3.4.6.1 If in the RF-I-START state, the initiating ADS-RF-ASE shall:

a) stop the t-RF-1 timer,

b) invoke ADS-start-forward confirmation with parameters as defined in Table 2.2.2.5-6, and

c) enter the RF-I-ACTIVE state.

Table 2.2.2.5-6

Parameter Name	Derivation of Parameter Value
Reply	"accepted"

2.2.2.5.3.4.7 Upon receipt of a D-START confirmation with a *Result* parameter value of "rejected (permanent)":

2.2.2.5.3.4.7.1 If in the RF-I-START state, the initiating ADS-RF-ASE shall:

a) stop the t-RF-1 timer,

b) invoke ADS-start-forward confirmation with parameters as defined in Table 2.2.2.5-7, and

c) enter the RF-I-IDLE state.

Table 2.2.2.5-7

Parameter Name	Derivation of Parameter Value
Reply	"Incompatible Version"

2.2.2.5.3.4.8 Upon receipt of a D-END confirmation with a *Result* parameter value of "accepted":

2.2.2.5.3.4.8.1 If in the RF-I-END state, the initiating ADS-RF-ASE shall:

a) stop the t-RF-2 timer, and

b) enter the RF-I-IDLE state.

2.2.2.5.3.4.9 Upon receipt of a D-END confirmation with a *Result* parameter value of "rejected":

2.2.2.5.3.4.9.1 If in the RF-I-END state, the initiating ADS-RF-ASE shall:

a) stop the t-RF-2 timer,

b) invoke D-ABORT request with parameters as defined in Table 2.2.2.5-8, and

c) enter the RF-I-IDLE state.

Table 2.2.2.5-8

Parameter Name	Derivation of Parameter Value
Originator	"provider"
User data	aDS-provider-abort-PDU with value dialogue-end-not-accepted

2.2.2.5.3.4.10 Upon receipt of a D-ABORT indication with the *Originator* parameter value set to "user":

2.2.2.5.3.4.10.1 If in the RF-I-START state or the RF-I-ACTIVE state, the initiating ADS-RF-ASE shall:

a) stop any timers,

b) invoke ADS-user-abort indication, and

c) enter the RF-I-IDLE state.

2.2.2.5.3.4.11 Upon receipt of a D-ABORT indication with the *Originator* parameter value set to "provider":

2.2.2.5.3.4.11.1 If in the RF-I-START state or the RF-I-ACTIVE state, the initiating ADS-RF-ASE shall:

a) stop any timers,

b) invoke ADS-provider-abort indication with parameter values as defined in Table 2.2.2.5-9, and

c) enter the RF-I-IDLE state.

Table 2.2.2.5-9

Parameter Name	Derivation of Parameter Value
Reason	D-ABORT *user data* parameter

2.2.2.5.3.4.12 Upon receipt of a D-P-ABORT indication:

2.2.2.5.3.4.12.1 If in the RF-I-START state or the RF-I-ACTIVE state, the initiating ADS-RF-ASE shall:

a) stop any timers,

b) invoke ADS-provider-abort indication with parameter values as defined in Table 2.2.2.5-10, and

c) enter the RF-I-IDLE state.

Table 2.2.2.5-10

Parameter Name	Derivation of Parameter Value
Reason	communications-service-failure

2.2.2.5.3.4.12.2 If in the RF-I-END state, the initiating ADS-RF-ASE shall:

a) stop any timers, and

b) enter the RF-I-IDLE state.

2.2.2.5.3.5 Responding ADS-RF-ASE

Note.— The responding ADS-RF-ASE has the following states:

a) RF-R-IDLE

b) RF-R-ACTIVE

2.2.2.5.3.5.1 On initiation, the responding ADS-RF-ASE shall be in the RF-R-IDLE state.

2.2.2.5.3.5.2 Upon receipt of an ADS-user-abort request:

2.2.2.5.3.5.2.1 If in the RF-I-ACTIVE state, the responding ADS-RF-ASE shall:

a) invoke D-ABORT request with parameters as defined in Table 2.2.2.5-11,

b) enter the RF-I-IDLE state.

Table 2.2.2.5-11

Parameter Name	Derivation of Parameter Value
Originator	“user”
User data	Not provided

2.2.2.5.3.5.3 Upon receipt of a D-START indication:

2.2.2.5.3.5.3.1 If in the RF-R-IDLE state, and if the D-START *DS-User Version Number* parameter is not compatible with the version number of the responding ADS-RF-ASE, and the *application service priority* parameter value is “high priority flight safety messages”, and the *RER quality of service* parameter is the abstract value “low”, it shall

a) invoke D-START response with parameter values as defined in Table 2.2.2.5-12, and

b) remain in the RF-R-IDLE state.

Table 2.2.2.5-12

Parameter Name	Derivation of Parameter Value
DS-user version number	The version number of the ADS-RF-ASE
Security requirements	Not provided
Quality of service	Not provided
Result	rejected (permanent)
User Data	Not provided

2.2.2.5.3.5.3.2 If in the RF-R-IDLE state, and if the D-START *DS-User Version Number* parameter is compatible with the version number of the responding ADS-RF-ASE, it shall:

a) invoke ADS-start-forward indication with parameter values as defined in Table 2.2.2.5-14,

b) invoke D-START response with parameter values as defined in Table 2.2.2.5-13, and

c) enter the RF-R-ACTIVE state.

Table 2.2.2.5-13

Parameter Name	Derivation of Parameter Value
DS-user version number	Not provided
Security requirements	Not provided
Quality of service	Not provided
Result	accepted
User Data	Not provided

Table 2.2.2.5-14

Parameter Name	Derivation of Parameter Value
Forwarded report details	D-START *user data* parameter, if provided

2.2.2.5.3.5.4 Upon receipt of a D-DATA indication containing an ADS-forwarded-report-PDU in the user data parameter:

2.2.2.5.3.5.4.1 If in the RF-R-ACTIVE state, the responding ADS-RF-ASE shall:

a) invoke ADS-forward-report indication with parameters as defined in Table 2.2.2.5-15, and

b) remain in the RF-R-ACTIVE state.

Table 2.2.2.5-15

Parameter Name	Derivation of Parameter Value
User data	D-DATA *user data* parameter

2.2.2.5.3.5.5 Upon receipt of a D-END indication:

2.2.2.5.3.5.5.1 If in the RF-R-ACTIVE state, the responding ADS-RF-ASE shall:

a) invoke ADS-end-forward indication,

b) invoke D-END response with parameters as defined in Table 2.2.2.5-16, and

c) enter the RF-R-IDLE state.

Table 2.2.2.5-16

Parameter Name	Derivation of Parameter Value
Result	“accepted”
User data	Not provided

2.2.2.5.3.5.6 Upon receipt of a D-ABORT indication with the *Originator* parameter value set to “user”:

2.2.2.5.3.5.6.1 If in the RF-R-ACTIVE state, the responding ADS-RF-ASE shall:

a) invoke ADS-user-abort indication, and

b) enter the RF-R-IDLE state.

2.2.2.5.3.5.7 Upon receipt of a D-ABORT indication with the *Originator* parameter value set to "provider":

2.2.2.5.3.5.7.1 If in the RF-R-ACTIVE state, the responding ADS-RF-ASE shall:

a) invoke ADS-provider-abort indication with parameter values as defined in Table 2.2.2.5-17, and

b) enter the RF-R-IDLE state.

Table 2.2.2.5-17

Parameter Name	Derivation of Parameter Value
Reason	D-ABORT *user data* parameter

2.2.2.5.3.5.8 Upon receipt of a D-P-ABORT indication:

2.2.2.5.3.5.8.1 If in the RF-R-ACTIVE state, the responding ADS-RF-ASE shall:

a) invoke ADS-provider-abort indication with parameter values as defined in Table 2.2.2.5-18, and

b) enter the RF-R-IDLE state.

Table 2.2.2.5-18

Parameter Name	Derivation of Parameter Value
Reason	aDS-provider-abort-PDU with value communications-service-failure

2.2.2.5.4 Exception Handling

2.2.2.5.4.1 Timer Expires

2.2.2.5.4.1.1 When either t-RF-1 or t-RF-2 timer expires, the ADS-RF-ASE shall :

a) invoke D-ABORT with Originator parameter value *DS user* and *user data* parameter value aDS-provider-abort-PDU with value *timer-expiry*,

b) if not in the ADS-I-IDLE state, invoke ADS-provider-abort indication with reason *timer-expiry*, and

c) enter the RF-I-IDLE state.

2.2.2.5.4.2 Unrecoverable System Error

2.2.2.5.4.2.1 **Recommendation**.— *When the ADS-RF-ASE has an unrecoverable system error, it should:*

a) invoke D-ABORT with Originator parameter value DS user and user data parameter value aDS-provider-abort-PDU with value unrecoverable-system-error,

b) if not in the ADS-R-IDLE state or ADS-I-IDLE state, invoke ADS-provider-abort indication with reason unrecoverable-system-error, and

c) if the initiator, enter the RF-I-IDLE state; if the responder, enter the RF-R-IDLE state.

2.2.2.5.4.3 Invalid PDU

2.2.2.5.4.3.1 When the user data parameter value of a D-START indication is a valid APDU and is not an aDS-forwarded-report-PDU, or the user data parameter value of a D-START confirmation is present, or the user data parameter value of a D-DATA indication is not an aDS-forwarded-report-PDU, or the user data parameter of a D-END indication is present, or the user data parameter of a D-END confirmation is present, the ADS-RF-ASE shall:

a) invoke D-ABORT with Originator parameter value *DS user* and *user data* parameter value aDS-provider-abort-PDU with value *invalid-PDU*,

b) if not in the ADS-R-IDLE state or ADS-I-IDLE state, invoke ADS-provider-abort indication with reason *invalid-PDU*, and

c) if the initiator, enter the RF-I-IDLE state; if the responder, enter the RF-R-IDLE state.

2.2.2.5.4.4 Sequence Error

2.2.2.5.4.4.1 When a PDU is delivered to the ADS-RF-ASE for which instructions are not stated in 2.2.2.5, it shall:

a) invoke D-ABORT with Originator parameter value *DS user* and *user data* parameter value *sequence-error*,

b) if not in the ADS-R-IDLE state or ADS-I-IDLE state, invoke ADS-provider-abort indication with reason *sequence-error*, and

c) if the initiator, enter the RF-I-IDLE state; if the responder, enter the RF-R-IDLE state.

2.2.2.5.4.5 D-START Rejection

2.2.2.5.4.5.1 Upon receipt of a D-START confirmation with the *result* parameter value containing the abstract value rejected (transient) or rejected (permanent), and the *reject source* parameter value containing the abstract value DS provider, the ADS-RF-ASE shall:

a) invoke ADS-provider-abort indication with reason *cannot-establish-contact*, and

b) enter the RF-I-IDLE state.

2.2.2.5.4.6 Decoding Error

2.2.2.5.4.6.1 When the ADS-RF-ASE fails to decode an APDU, it shall

a) invoke D-ABORT with *Originator* parameter value *DS user* and *user data* parameter value aDS-provider-abort-PDU with value *decoding-error*, and

b) if not in the ADS-R-IDLE state, invoke ADS-provider-abort indication with reason *decoding-error*, and

c) enter the RF-I-IDLE state.

2.2.2.5.4.7 Invalid QOS

2.2.2.5.4.7.1 Upon receipt of a D-START indication with the *application service priority* parameter set to a value other than the abstract value “high priority flight safety messages”, or the *RER quality of service* parameter set to a value other then the abstract value “low”, the ADS-RF-ASE shall:

a) invoke D-ABORT with Originator parameter value DS user and user data parameter value aDS-provider-abort-PDU with value invalid-qos-parameter; and

b) enter the RF-R-IDLE state.

2.2.2.5.5 ADS-ASE State Tables

2.2.2.5.5.1 Priority

2.2.2.5.5.1.1 If the state tables for the ADS-RF-ASE shown below conflict with textual statements made elsewhere in this document, the textual statements shall take precedence.

Note 1.— In the following state tables, the statement “cannot occur” means that if the implementation conforms to the SARPs, it is impossible for this event to occur. If the event does occur, this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE aborts with the error “internal system error”.

Note 2.— In the following state tables, the statement "not permitted" means that the implementation must prevent this event from occurring through some local means. If the event does occur this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE performs a local rejection of the request rather than aborting the dialogue.

Table 2.2.2.5-19. Initiating RF ASE state table

State ⇒ *Event* ⇓	*RF-I-IDLE* (Initial State)	*RF-I-START*	*RF-I-ACTIVE*	*RF-I-END*
Primitive Requests and Responses				
ADS-start-forward req	D-START req start t-RF-1 *RF-I-START*	Not permitted	Not permitted	Not permitted
ADS-forward-report req	Not permitted	Not permitted	D-DATA req *RF-I-ACTIVE*	Not permitted
ADS-end-forward req	Not permitted	Not permitted	D-END req start t-RF-2 *RF-I-END*	Not permitted
ADS-user-abort req	Not permitted	D-ABORT req *RF-I-IDLE*	D-ABORT req *RF-I-IDLE*	D-ABORT req *RF-I-IDLE*
Primitive Indications and Confirmations				
D-START cnf accepted	Cannot occur	stop t-RF-1 ADS-start-forward cnf + *RF-I-ACTIVE*	Cannot occur	Cannot occur
D-START cnf rejected	Cannot occur	stop t-RF-1 ADS-start-forward cnf - RF-I-IDLE	Cannot occur	Cannot occur
D-END cnf accepted	Cannot occur	Cannot occur	Cannot occur	stop t-RF-2 *RF-I-IDLE*
D-END cnf rejected	Cannot occur	Cannot occur	Cannot occur	stop t-RF-2 D-ABORT req *RF-I-IDLE*
Timer Expiry				
t-RF-1	Cannot occur	D-ABORT req ADS-provider-abort ind *RF-I-IDLE*	Cannot occur	Cannot occur
t-RF-2	Cannot occur	Cannot occur	Cannot occur	D-ABORT req *RF-I-IDLE*

Table 2.2.2.5-20. Responding RF ASE state table

State ⇒ *Event* ⇓	*RF-R-IDLE* (Initial State)	*RF-R-ACTIVE*
Primitive Requests and Responses		
ADS-user-abort	Not permitted	D-ABORT req *RF-R-IDLE*
Primitive Indications and Confirmations		
D-START ind with compatible version number	D-START rsp + ADS-start-forward ind *RF-R-ACTIVE*	Cannot occur
D-START ind with incompatible version number	D-START rsp - *RF-R-IDLE*	Cannot occur
D-DATA ind	Cannot occur	ADS-forward-report ind *RF-R-ACTIVE*
D-END ind	Cannot occur	ADS-end-forward ind D-END rsp + *RF-R-IDLE*
D-ABORT ind with originator="user"	Cannot occur	ADS-user-abort ind *RF-R-IDLE*
D-ABORT ind with originator="provider" or D-P-ABORT ind	Cannot occur	ADS-provider-abort ind *RF-R-IDLE*

2.2.2.6 Communication Requirements

2.2.2.6.1 Encoding Rules

2.2.2.6.1.1 The ADS application shall use PER as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in 2.2.2.4.

2.2.2.6.2 Dialogue Service Requirements

2.2.2.6.2.1 Primitive Requirements

2.2.2.6.2.1.1 Where dialogue service primitives, that is D-START, D-END, D-ABORT, D-P-ABORT and D-DATA are described as being invoked in 2.2.2.5, the ADS-ground-ASE and the ADS-air-ASE shall exhibit external behavior consistent with the dialogue service, as described in 4.2, having been implemented and its primitives invoked.

2.2.2.6.2.2 Quality of Service Requirements

2.2.2.6.2.2.1 The application service priority for ADS shall have the abstract value of "high priority flight safety messages".

2.2.2.6.2.2.2 The *RER quality of service* parameter of the D-START request shall be set to the abstract value of "low".

2.2.2.6.2.2.3 The ADS-ASE shall map the class of communication service abstract values to the ATSC routing class abstract value part of the D-START QOS parameter as presented in Table 2.2.2.6-1.

Table 2.2.2.6-1. Mapping between class of communication and routing class abstract values

Class of Communication Abstract Value	Routing Class Abstract Value
A	Traffic follows Class A ATSC route(s)
B	Traffic follows Class B ATSC route(s)
C	Traffic follows Class C ATSC route(s)
D	Traffic follows Class D ATSC route(s)
E	Traffic follows Class E ATSC route(s)
F	Traffic follows Class F ATSC route(s)
G	Traffic follows Class G ATSC route(s)
H	Traffic follows Class H ATSC route(s)

Note.— ATSC values are defined in 1.3.

2.2.2.7 ADS Report Forwarding User Requirements

2.2.2.7.1 Establishment and operation of a Forward Contract

Note 1.— 2.2.2.7.1 details the actions taken by an ADS-RF-user and a second ADS-RF-user when the first forwards ADS reports to the second.

Note 2.— When an ADS-RF-user requires to initiate forwarding of ADS-reports, it initiates ADS-start-forward request.

Note 3.— When the initiating ADS-RF-user receives an ADS-start-forward confirmation with the Reply parameter value set to "accepted", it invokes ADS-forward-report request when it requires to forward ADS-reports.

Note 4.— When the initiating ADS-RF-user requires to stop forwarding ADS-reports, it invokes ADS-end-forward request.

2.2.2.7.2 Operation of Aborts

Note 1.— 2.2.2.7.2 details the actions taken by an ADS-RF-user when aborts occur.

Note 2.— When an ADS-RF-user requires to abort the current contract, it invokes ADS-user-abort request.

2.2.2.7.3 Parameter Value Unit, Range and Resolution

2.2.2.7.3.1 An ADS Report Forwarding user shall interpret ADS Report Forwarding parameter value unit, range and resolution as defined in 2.2.2.4.

2.2.2.8 Subsetting Rules

2.2.2.8.1 General

Note.— 2.2.2.8 specifies conformance requirements which all implementations of the ADS Report Forwarding protocol obey.

2.2.2.8.1.1 An implementation of the ADS Report Forwarding service claiming conformance to 2.2.2 shall support the ADS Report Forwarding protocol features as shown in the tables below.

Note.— The 'status' column indicates the level of support required for conformance to the ARF-ASE protocol described in this 2.2.2. The values are as follows:

a) ***'M'** mandatory support is required,*

b) ***'O'** optional support is permitted for conformance to the ADS Report Forwarding protocol,*

c) ***'N/A'** the item is not applicable, and*

d) ***'C.n'** the item is conditional where n is the number which identifies the condition which is applicable.*

Table 2.2.2.8-1. ADS Report Forwarding Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 2.2.2.8-2. ARF Protocol Options

	Status	**Associated Predicate**
ARF initiator	O	INIT

Table 2.2.1.8-3. ADS-ground-ARF Conformant Configurations

	List of Predicates	**Functionality Description**
I	INIT	ASE operating as ARF initiator or ARF receiver
II	none	ASE operating as ARF receiver only

Table 2.2.2.8-4. Supported ADS Report Forwarding Service Primitives

	Sending (req,[cnf])	**Receiving (ind, [rsp])**
ARF-start-forward	if (INIT) M, else N/A	M
ARF-forward-report	if (INIT) M, else N/A	M
ARF-end-forward	if (INIT) M, else N/A	M
ARF-user-abort	M	M
ARF-provider-abort	N/A	M

Table 2.2.2.8-5. Supported ARF APDUs

	Sender	**Receiver**
[ADSRFPDUs] aDS-forwarded-report-PDU	if (INIT) M, else N/A	M
[ADSRFPDUs] aDS-provider-abort-PDU	M	M

2.3 CONTROLLER PILOT DATA LINK COMMUNICATION APPLICATION

2.3.1 INTRODUCTION

2.3.1.1 Overview

2.3.1.1.1 The CPDLC application allows data link communication between controllers and pilots.

2.3.1.1.2 The CPDLC application provides the capability to establish, manage, and terminate CPDLC dialogues between ATS ground and aircraft system peers. Once a dialogue is established, CPDLC provides for controller/pilot message exchange.

2.3.1.1.3 The CPDLC application also provides the capability to establish, manage, and terminate CPDLC dialogues between two ATC ground system peers for the purpose of ground/ground forwarding of a CPDLC message.

Note 1.— Structure:

a) 2.3.1: INTRODUCTION contains the structure of 2.3 and a summary of the functional capabilities of CPDLC.

b) 2.3.2: GENERAL REQUIREMENTS contains the CPDLC version number, and error processing requirements.

c) 2.3.3: ABSTRACT SERVICE DEFINITION contains the description of the abstract service provided by the CPDLC Application Service Element (CPDLC-ASE).

d) 2.3.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by CPDLC ASEs using Abstract Syntax Notation Number One (ASN.1).

e) 2.3.5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the CPDLC protocol, as well as time constraints and CPDLC-ASE protocol descriptions and state tables.

f) 2.3.6: COMMUNICATION REQUIREMENTS contains the requirements that the CPDLC application imposes on the underlying communication system.

g) 2.3.7: CPDLC USER REQUIREMENTS contains requirements imposed on the user of the CPDLC ASE service and message description tables.

h) 2.3.8: SUBSETTING RULES defines the conformant subsets for the CPDLC-ASE.

Note 2.— Functional Descriptions

*a) The **Controller-Pilot Message Exchange Function** defines a method for a controller and pilot to exchange messages via data link. This function provides messages for the following :*

1) general information exchange;

2) clearance

i) delivery,

ii) request, and

iii) response;

3) altitude/identity surveillance;

4) monitoring of current/planned position;

5) advisories

i) request and

ii) delivery;

6) system management functions; and

7) emergency situations.

*b) The **Transfer of Data Authority Function** provides the capability for the current data authority to designate another ground system as the next data authority. A CPDLC dialogue can be opened with or by the next data authority at a time before becoming the current data authority. This capability is intended to prevent a loss of communication that would occur if the next data authority were prevented from actually setting up a dialogue with an aircraft until it became the current data authority. The designation of a next data authority is accomplished using a CPDLC message.*

*c) The **Down Stream Clearance Function** provides the capability for an aircraft to contact an air traffic service unit which is not the current data authority for the purpose of receiving a down stream clearance. This information is exchanged using CPDLC message(s).*

*d) The **Ground Forward Function** provides the capability for a ground system to forward information received in a CPDLC message to another ground system. The*

ground forwarding function can be used by the controlling data authority to forward an aircraft request to the next data authority, so that an aircraft does not need to issue the same request again. This function can also be used by a downstream data authority to pass a message to a current data authority for transmission by the current data authority to an aircraft. This information is exchanged using CPDLC message(s). It is a one-way forwarding of information with an indication of success, failure or non-support from the receiving ground system.

Note 3.— See 2.3.7 for detailed CPDLC message intent/use descriptions.

2.3.2 GENERAL REQUIREMENTS

2.3.2.1 CPDLC ASE Version Number

2.3.2.1.1 The CPDLC-air-ASE and CPDLC-ground-ASE version numbers shall both be set to one.

2.3.2.2 Error Processing Requirements

2.3.2.2.1 In the event of information input by the CPDLC-user being incompatible with that able to be processed by the system, the CPDLC-user shall be notified.

2.3.2.2.2 In the event of a CPDLC-user invoking a CPDLC service primitive when the CPDLC-ASE is not in a state specified in 2.3.5, the CPDLC-user shall be notified.

2.3.3 THE ABSTRACT SERVICE

2.3.3.1 Service Description

2.3.3.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service shall exhibit external behavior consistent with having implemented a CPDLC-ground-ASE, or CPDLC-air-ASE respectively, with the following abstract service interface primitives, making them available to the CPDLC-ground-user or CPDLC-air-user respectively.

Note 1.— There is no requirement to implement the service in a CPDLC product; however, it is necessary to implement the ground based and air based system in such a way that it will be impossible to detect (from the peer system) whether or not the interface has been built.

Note 2.— This chapter defines the abstract service interface for the CPDLC service. The CPDLC-ASE abstract service is described in this chapter from the viewpoint of the CPDLC-air-user, the CPDLC-ground-user and the CPDLC-service-provider.

Note 3.— This chapter defines the static behavior (i.e., the format) of the CPDLC abstract service. Its dynamic behavior (i.e., how it is used) is described in 2.3.7.

Note 4.— Figure 2.3.3-1 shows the functional model of the CPDLC Application. The functional modules identified in this model are the following :

a) the CPDLC-user,

b) the CPDLC Application Entity (CPDLC-AE) service,

c) the CPDLC-AE,

d) the CPDLC Control Function (CPDLC-CF),

e) the CPDLC Application Service Element (CPDLC-ASE) service,

f) the CPDLC-ASE, and

g) the Dialogue Service (DS).

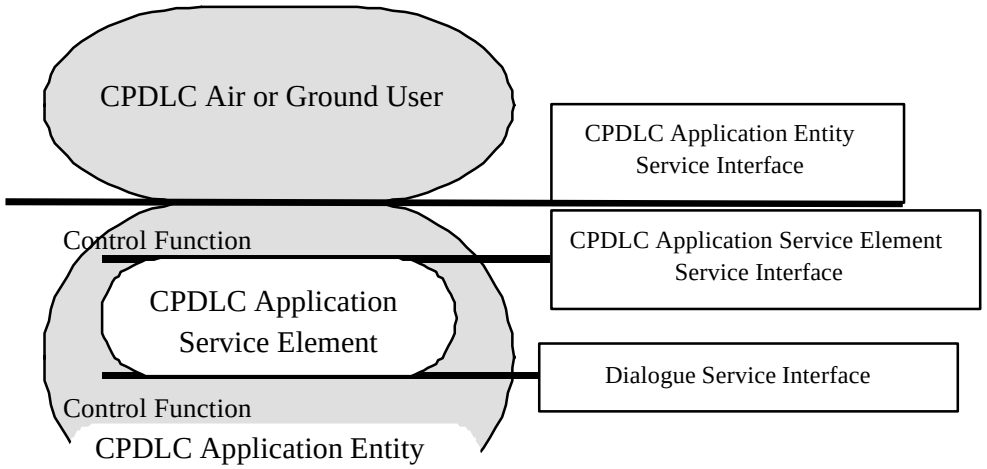

Figure 2.3.3-1. Functional Model of the CPDLC Application

Note 5.— The CPDLC-user represents the operational part of the CPDLC system. This user does not perform the communication functions but relies on a communication service provided to it via the CPDLC-AE through the CPDLC-AE service. The individual actions possible through the CPDLC-AE service are called service primitives.

Note 6.— The CPDLC-AE consists of several elements including the CPDLC-ASE and the CPDLC-CF.

Note 7.— The CPDLC-ASE is the element in the communication system which executes the CPDLC specific protocol. In other words, it takes care of the CPDLC specific service primitive sequencing, message creation, timer management, error and exception handling.

Note 8.— This CPDLC-CF is responsible for mapping service primitives received from one element (such as the CPDLC-ASE and the CPDLC-user) to service primitives of other abstract elements.

Note 9.— The CPDLC-ASE has two abstract boundaries with the CPDLC-CF: the CPDLC-ASE service and the dialogue service. The CPDLC-CF maps CPDLC-AE service primitives to other abstract elements in the CPDLC-AE and the underlying communication service, and vice versa.

2.3.3.2 The CPDLC-ASE Abstract Service

2.3.3.2.1 The CPDLC-ASE abstract service shall consist of a subset of the following services as allowed in 2.3.8:

a) CPDLC-start service as defined in 2.3.3.3,

b) DSC-start service as defined in 2.3.3.4,

c) CPDLC-message service as defined in 2.3.3.5,

d) CPDLC-end service as defined in 2.3.3.6,

e) DSC-end service as defined in 2.3.3.7,

f) CPDLC-forward service as defined in 2.3.3.8,

g) CPDLC-user-abort service as defined in 2.3.3.9, and

h) CPDLC-provider-abort service as defined in 2.3.3.10.

Note 1.— For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables in 2.3.3.

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 left being present, and equal to that value;*
d)	***M***	*mandatory;*
e)	***M(=)***	*mandatory, and equal to the value of the parameter to the left;*
f)	***U***	*user option.*

Note 2.— The following abbreviations are used in this document:

a) ***Req*** *- request; data is input by CPDLC-user initiating the service to its respective ASE,*

b) ***Ind*** *- indication; data is indicated by the receiving ASE to its respective CPDLC-user,*

c) ***Rsp*** *- response; data is input by receiving CPDLC user to its respective ASE, and*

d) ***Cnf*** *- confirmation; data is confirmed by the initiating ASE to its respective CPDLC-user.*

Note 3.— An unconfirmed service allows a message to be transmitted in one direction without providing a corresponding response.

Note 4.— A confirmed service provides end-to-end confirmation that a message sent by one user was received by its peer user.

Note 5.— An abstract syntax is a syntactical description of a parameter which does not imply a specific implementation. Only when the CPDLC-ASE maps a parameter into an APDU field, or vice versa, the abstract syntax of the parameter is described by using ASN.1 of 2.3.4 for this field.

2.3.3.3 CPDLC-start Service

Note 1.— The CPDLC-start service is used by the CPDLC-air-user or CPDLC-ground-user to establish a CPDLC dialogue. It is a confirmed service.

Note 2.— Once a CPDLC dialogue is established it remains open until explicitly closed. (See CPDLC-end and CPDLC-abort services.)

2.3.3.3.1 The CPDLC-start service shall contain the primitives and parameters as presented in Table 2.3.3-1.

Table 2.3.3-1. CPDLC-start Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Called Peer Identifier	M			
Calling Peer Identifier	M	M(=)		
CPDLC Message	U	C(=)		
Reject Reason			C	C(=)
Result			M	M(=)
Class of Communication Service	U	M		

2.3.3.3.2 Called Peer Identifier

Note 1.— If the service is ground initiated, this parameter contains the addressed aircraft's 24-bit aircraft address.

Note 2.— If the service is air initiated, this parameter contains the addressed ground system's facility designation.

2.3.3.3.2.1 If the service is ground initiated, the *Called Peer Identifier* parameter value shall conform to the abstract syntax 24-bit aircraft address.

2.3.3.3.2.2 If the service is air initiated, the *Called Peer Identifier* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.3.3.3.3 Calling Peer Identifier

Note 1.— If the service is ground initiated, this parameter contains the sending ground system's facility designation.

Note 2.— If the service is air initiated, this parameter contains the sending aircraft's 24-bit aircraft address.

2.3.3.3.3.1 If the service is ground initiated, the *Calling Peer Identifier* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.3.3.3.3.2 If the service is air initiated, the *Calling Peer Identifier* parameter value shall conform to the abstract syntax 24-bit aircraft address.

2.3.3.3.4 CPDLC Message

Note.— The CPDLC-user can use this parameter to send a CPDLC message to its peer user.

2.3.3.3.4.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage, if supplied by the CPDLC-ground-user.

2.3.3.3.4.2 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage, if supplied by the CPDLC-air-user.

2.3.3.3.5 Reject Reason

Note.— This parameter is used to provide a reason for rejecting a CPDLC dialogue.

2.3.3.3.5.1 If the CPDLC-user accepts the request to open a CPDLC dialogue, the CPDLC user shall be prohibited from providing a CPDLC message for the *Reject Reason* parameter.

2.3.3.3.5.2 The *Reject Reason* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage if supplied by the CPDLC-ground-user.

2.3.3.3.5.3 The *Reject Reason* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage if supplied by the CPDLC-air-user.

2.3.3.3.6 Result

Note.— This parameter is used to indicate whether or not a requested CPDLC dialogue is accepted.

2.3.3.3.6.1 This parameter shall have one of two abstract values: "accepted" or "rejected".

2.3.3.3.7 Class of Communication Service

Note 1.— This parameter contains the value of the required class of communication service. If not specified by the CPDLC-user, this indicates that there is no routing preference.

Note 2.— This parameter is used by the CPDLC-ground-user to determine if the Class of Communication value is acceptable for the establishment of a CPDLC dialogue.

Note 3.— The parameter indicated to to the peer user is that provided by the user if specified by the user, else the parameter indicated to the peer user is that whta the dialogue provider service supplies.

2.3.3.3.7.1 Where specified by the CPDLC-user, the *Class of Communication Service* parameter shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G”, or “H”.

2.3.3.3.7.2 When this parameter is provided by the user, the same value shall be indicated to the peer user.

2.3.3.4 DSC-start Service

Note 1.— The DSC-start service is used to establish a DSC dialogue for the purpose of providing down stream clearances. It is a confirmed service.

Note 2.— Once a DSC dialogue is established it remains open until explicitly closed. (See DSC-end and CPDLC-abort services.)

2.3.3.4.1 The DSC-start service shall contain the primitives and parameters as presented in Table 2.3.3-2.

Table 2.3.3-2. DSC-start Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Facility Designation	M			
Aircraft Address	M	M(=)		
CPDLC Message	U	C(=)		
Reject Reason			C	C(=)
Result			M	M(=)
Class of Communication Service	U	M		

2.3.3.4.2 Facility Designation

Note.— This parameter contains the addressed ground system’s facility designation.

2.3.3.4.2.1 The *Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.3.3.4.3 Aircraft Address

2.3.3.4.3.1 The *Aircraft Address* parameter value shall conform to the abstract syntax 24-bit aircraft address.

Note.— This parameter contains the aircraft's 24 bit aircraft address.

2.3.3.4.4 CPDLC Message

Note.— The CPDLC-air-user can use this parameter to send a CPDLC message to a CPDLC-ground-user.

2.3.3.4.4.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage.

2.3.3.4.5 Reject Reason

Note.— The parameter is used to provide a reason for rejecting a DSC dialogue.

2.3.3.4.5.1 If, the CPDLC-ground-user accepts the request to open a DSC dialogue, the CPDLC-ground-user shall be prohibited from providing a CPDLC message for the *Reject Reason* parameter.

2.3.3.4.5.2 The *Reject Reason* parameter shall conform to the ASN.1 abstract syntax ATCUplinkMessage.

2.3.3.4.6 Result

Note.— This parameter is used to indicate whether or not a requested DSC dialogue is accepted.

2.3.3.4.6.1 The *Result* parameter value shall have one of two abstract values: "accepted" or "rejected".

2.3.3.4.7 Class of Communication Service

Note 1.— This parameter contains the value of the required class of communication service. If not specified by the CPDLC-air-user, this indicates that there is no routing preference.

Note 2.— This parameter is used by the CPDLC-ground-user to determine if the Class of Communication value is acceptable for the establishment of a CPDLC dialogue.

Note 3.— If provided by the user, the parameter indicated to the peer user is that provided by the user, else it is what the dialogue provider service supplies.

2.3.3.4.7.1 Where specified by the CPDLC-air-user, the *Class of Communication Service* parameter shall have one of the following abstract values: "A", "B", "C", "D", "E", "F", "G", or "H".

2.3.3.4.7.2 When this parameter is provided by the user, the same value shall be indicated to the peer user.

2.3.3.5 CPDLC-message Service

Note.— The CPDLC-message service can be used for pilot/controller message exchange, once a dialogue is established. It is an unconfirmed service.

2.3.3.5.1 The CPDLC-message service shall contain the primitives and parameters as presented in Table 2.3.3-3.

Table 2.3.3-3. CPDLC-message Service Parameters

Parameter Name	Req	Ind
CPDLC Message	M	M(=)

2.3.3.5.2 CPDLC Message

Note.— This parameter contains a CPDLC message.

2.3.3.5.2.1 If the CPDLC-message service is invoked by the CPDLC-ground-user, the *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage.

2.3.3.5.2.2 If the CPDLC-message service is invoked by the CPDLC-air-user, the *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage.

2.3.3.6 CPDLC-end Service

Note.— The CPDLC-end service is used by the CPDLC-ground-user to end a CPDLC dialogue with a CPDLC-air-user. It is a confirmed service.

2.3.3.6.1 The CPDLC-end service shall contain the primitives and parameters as presented in Table 2.3.3-4.

Table 2.3.3-4. CPDLC-end Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
CPDLC Message	U	C(=)	U	C(=)
Result			M	M(=)

2.3.3.6.2 CPDLC Message

Note.— This parameter contains a CPDLC message.

2.3.3.6.2.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage, if provided by the CPDLC-ground-user.

2.3.3.6.2.2 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage, if provided by the CPDLC-air-user.

2.3.3.6.3 Result

Note.— This parameter is used to indicate whether or not a request to terminate a CPDLC dialogue is accepted.

2.3.3.6.3.1 The *Result* parameter shall have one of two abstract values: “accepted” or “rejected”.

2.3.3.7 DSC-end Service

Note.— The DSC-end service is used by the DSC-air-user to end a DSC dialogue with a CPDLC-ground-user. It is a confirmed service.

2.3.3.7.1 The DSC-end service shall contain the primitives and parameters as presented in Table 2.3.3-5.

Table 2.3.3-5. DSC-end Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
CPDLC Message	U	C(=)	U	C(=)
Result			M	M(=)

2.3.3.7.2 CPDLC Message

Note.— This parameter contains a CPDLC message.

2.3.3.7.2.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage, if provided by the CPDLC-ground-user.

2.3.3.7.2.2 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage if provided by the CPDLC-air-user.

2.3.3.7.3 Result

Note.— This parameter is used to indicate whether or not a request to terminate a DSC dialogue is accepted.

2.3.3.7.3.1 The *Result* parameter shall have one of two abstract values: “accepted” or “rejected”.

2.3.3.8 CPDLC-forward Service

Note.— The CPDLC-forward service is used by a CPDLC-ground-user to send a CPDLC message to another CPDLC-ground-user. Its primary use is for the forwarding of aircraft requests.

2.3.3.8.1 If the CPDLC-forward service is supported by the receiving ground system and the sending CPDLC-ground-ASE and receiving CPDLC-ground-ASE version numbers are equal, the CPDLC-forward service shall contain the primitives and parameters as presented in Table 2.3.3-6.

Table 2.3.3-6. CPDLC-forward Service Parameters (Service Supported, Versions Equal)

Parameter Name	Req	Ind	Cnf
Called Facility Designation	M		
Calling Facility Designation	M	M(=)	
CPDLC Message	M	M(=)	
Class of Communication Service	U		
Result			M

2.3.3.8.2 If the CPDLC-forward service is not supported by the receiving ground system, or if the CPDLC-forward service is supported by the receiving ground system but the sending CPDLC-ground-ASE and receiving CPDLC-ground-ASE version numbers are not equal, the CPDLC-forward service shall contain the primitives and parameters as presented in Table 2.3.3-7.

Table 2.3.3-7. CPDLC-forward Service Parameters (Service Not Supported or Versions Not Equal)

Parameter Name	Req	Cnf
Called Facility Designation	M	
Calling Facility Designation	M	
ASE Version Number		C
CPDLC Message	M	
Class of Communication Service	U	
Result		M

2.3.3.8.3 Called Facility Designation

Note.— This parameter contains the addressed ground system's facility designation.

2.3.3.8.3.1 The *Called Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.3.3.8.4 Calling Facility Designation

Note.— This parameter contains the sending ground system's facility designation.

2.3.3.8.4.1 The *Calling Facility Designation* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.3.3.8.5 ASE Version Number

Note.— This parameter contains the version number of the CPDLC-ASE.

2.3.3.8.5.1 When provided by the CPDLC-ground-ASE, the *ASE Version Number* parameter shall conform to the abstract integer value in the range 1-255.

2.3.3.8.5.2 Only if the sending CPDLC-ground-ASE version number is not equal to the receiving CPDLC-ground-ASE version number shall the receiving CPDLC-ground-ASE version number be confirmed to the sending CPDLC-ground-user.

Note.— If the sending CPDLC-ground-ASE version number is the same as the receiving CPDLC-ground-ASE version number, the Version Number parameter is not present in the indication given to the receiving CPDLC-ground-user, nor in the confirmation to the sending CPDLC-ground-user.

2.3.3.8.6 CPDLC Message

Note.— The sending CPDLC-ground-user uses this parameter to forward a CPDLC message to another CPDLC-ground-user.

2.3.3.8.6.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCForwardMessage, when supplied by the CPDLC-ground-user.

2.3.3.8.7 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service. If not specified by the CPDLC-ground-user, this indicates that there is no routing preference.

2.3.3.8.7.1 Where specified by the CPDLC-ground-user, the *Class of Communication Service* parameter shall have one of the following abstract values: "A", "B", "C", "D", "E", "F", "G", or "H".

2.3.3.8.8 Result

Note.— This parameter contains the result of the CPDLC-forward service. It will indicate success (service supported and matching versions), service unsupported, or version number incompatibility.

2.3.3.8.8.1 The *Result* parameter value shall conform to the ASN.1 abstract syntax ATCForwardResponse.

2.3.3.9 CPDLC-user-abort Service

Note 1.— This service provides the capability for either the CPDLC-air-user or a CPDLC-ground-user to abort communication with its peer. It can be invoked at any time the user is aware that the CPDLC service is in operation. The CPDLC-user-abort service can be used for operational or technical reasons. It is an unconfirmed service. Messages in transit may be lost during this operation.

Note 2.— If the service is invoked prior to complete establishment of the dialogue, the CPDLC-user-abort indication may not be provided. A CPDLC-provider-abort indication may result instead.

2.3.3.9.1 The CPDLC-user-abort service shall contain the primitives and parameters as presented in Table 2.3.3-8.

Table 2.3.3-8. CPDLC-user-abort Service Parameters

Parameter Name	Req	Ind
Reason	U	M

2.3.3.9.2 Reason

Note 1.— This parameter is used to indicate a reason for aborting the CPDLC or DSC dialogue.

Note 2.— If provided by the user, the parameter indicated to the peer user is that provided by the user, else it is what the ASE supplies.

2.3.3.9.2.1 The *Reason* parameter value shall conform to the ASN.1 abstract syntax CPDLCUserAbortReason.

2.3.3.9.2.2 When this parameter is provided by the user, the same value shall be indicated to the peer user.

2.3.3.10 CPDLC-provider-abort Service

Note.— This service provides the capability for the CPDLC-service provider to inform its active users that it can no longer provide the CPDLC service. Messages in transit may be lost during this operation.

2.3.3.10.1 The CPDLC-provider-abort service shall contain the primitives and parameters as presented in Table 2.3.3-9.

Table 2.3.3-9. CPDLC-provider-abort Service Parameters

Parameter Name	Ind
Reason	M

2.3.3.10.2 Reason

Note.— This parameter identifies the reason for the abort.

2.3.3.10.2.1 The *Reason* parameter shall conform to the ASN.1 abstract syntax CPDLCProviderAbortReason.

2.3.4 FORMAL DEFINITIONS OF MESSAGES

2.3.4.1 Encoding/Decoding Rules

2.3.4.1.1 A CPDLC-air-ASE shall be capable of encoding AircraftPDUs APDUs and decoding GroundPDUs APDUs.

2.3.4.1.2 A CPDLC-ground-ASE shall be capable of encoding GroundPDUs APDUs and decoding AircraftPDUs APDUs.

2.3.4.2 CPDLC ASN.1 Abstract Syntax

2.3.4.2.1 The abstract syntax of the CPDLC protocol data units shall comply with the description contained in the ASN.1 module CPDLCMessageSetVersion1 conforming to ISO/IEC 8824, as defined in this section.

```
CPDLCMessageSetVersion1 DEFINITIONS::=

BEGIN

-- ------------------------------------------------------------------------
-- Ground Generated Messages - Top level
-- ------------------------------------------------------------------------
GroundPDUs ::= CHOICE
        {
        abortUser               [0]     CPDLCUserAbortReason,
        abortProvider           [1]     CPDLCProviderAbortReason,
        startup                 [2]     UplinkMessage,
        send                    [3]     ATCUplinkMessage,
        forward                 [4]     ATCForwardMessage,
        forwardresponse         [5]     ATCForwardResponse,
        ...
        }

UplinkMessage ::= CHOICE
        {
        noMessage               [0]     NULL,
        aTCUplinkMessage        [1]     ATCUplinkMessage
        }
```

```
ATCUplinkMessage ::= SEQUENCE
        {
        header                  ATCMessageHeader,
        messageData             ATCUplinkMessageData
        }

ATCUplinkMessageData ::= SEQUENCE
        {
        elementIds              SEQUENCE SIZE (1..5) OF ATCUplinkMsgElementId,
        constrainedData         SEQUENCE
                {
                routeClearanceData      SEQUENCE SIZE (1..2) OF RouteClearance OPTIONAL,
                ...
                }
                OPTIONAL
        }

ATCForwardMessage ::= SEQUENCE
        {
        forwardHeader           ForwardHeader,
        forwardMessage          ForwardMessage
        }

ForwardHeader ::= SEQUENCE
        {
        dateTime                DateTimeGroup,
        aircraftID              AircraftFlightIdentification,
        aircraftAddress         AircraftAddress
        }

ForwardMessage ::= CHOICE
        {
        upElementIDs            [0]     ATCUplinkMessageData,
        downElementIDs          [1]     ATCDownlinkMessageData
        }

ATCForwardResponse ::= ENUMERATED
        {
        success                         (0),
        service-not-supported           (1),
        version-not-equal               (2),
        ...
        }
```

```
-- ------------------------------------------------------------------------------
-- Aircraft Generated Messages - Top level
-- ------------------------------------------------------------------------------

AircraftPDUs::= CHOICE
        {
        abortUser       [0]     CPDLCUserAbortReason,
        abortProvider   [1]     CPDLCProviderAbortReason,
        startdown       [2]     StartDownMessage,
        send            [3]     ATCDownlinkMessage,
        ...
        }

StartDownMessage ::= SEQUENCE
        {
        mode                    Mode DEFAULT cpdlc,
        startDownlinkMessage    DownlinkMessage
        }

Mode ::= ENUMERATED
        {
        cpdlc           (0),
        dsc             (1)
        }

DownlinkMessage ::= CHOICE
        {
        noMessage                       [0]     NULL,
        aTCDownlinkMessage              [1]     ATCDownlinkMessage
        }

ATCDownlinkMessage ::= SEQUENCE
        {
        header                  ATCMessageHeader,
        messageData             ATCDownlinkMessageData
        }
```

```
ATCDownlinkMessageData ::= SEQUENCE
        {
        elementIds              SEQUENCE SIZE (1..5) OF ATCDownlinkMsgElementId,
        constrainedData         SEQUENCE
                {
                routeClearanceData      SEQUENCE SIZE (1..2) OF RouteClearance OPTIONAL,
                ...
                }
                OPTIONAL
        }

-- ------------------------------------------------------------------------------
-- Uplink and Downlink messages - Common Elements
-- ------------------------------------------------------------------------------

ATCMessageHeader ::= SEQUENCE
        {
        messageIdNumber         [0]     MsgIdentificationNumber,
        messageRefNumber        [1]     MsgReferenceNumber      OPTIONAL,
        dateTime                [2]     DateTimeGroup,
        logicalAck              [3]     LogicalAck              DEFAULT notRequired
        }

MsgIdentificationNumber ::= INTEGER (0..63)

MsgReferenceNumber ::= INTEGER (0..63)

CPDLCUserAbortReason ::= ENUMERATED
        {
        undefined                                       (0),
        no-message-identification-numbers-available     (1),
        duplicate-message-identification-numbers        (2),
        no-longer-next-data-authority                   (3),
        current-data-authority-abort                    (4),
        commanded-termination                   (5),
        invalid-response                                (6),
        ...
        }

CPDLCProviderAbortReason ::= ENUMERATED
        {
        timer-expired                                   (0),
        undefined-error                 (1),
        invalid-PDU                                     (2),
        not-permitted-PDU                               (3),
        communication-service-error                     (4),
```

```
        communication-service-failure       (5),
        invalid-QOS-parameter               (6),
        expected-PDU-missing                (7),
        ...
        }

LogicalAck ::= ENUMERATED
        {
        required                    (0),
        notRequired                 (1)
        }
-- ------------------------------------------------------------------------------
-- Uplink message element
-- ------------------------------------------------------------------------------

ATCUplinkMsgElementId ::= CHOICE
        {
--      UNABLE                                  Urg(N)/Alr(L)/Resp(N)
        uM0NULL                                 [0] NULL,

--      STANDBY                                 Urg(N)/Alr(L)/Resp(N)
        uM1NULL                                 [1] NULL,

--      REQUEST DEFERRED                        Urg(N)/Alr(L)/Resp(N)
        uM2NULL                                 [2] NULL,

--      ROGER                                   Urg(N)/Alr(L)/Resp(N-)
        uM3NULL                                 [3] NULL,

--      AFFIRM                                  Urg(N)/Alr(L)/Resp(N-)
        uM4NULL                                 [4] NULL,

--      NEGATIVE                                Urg(N)/Alr(L)/Resp(N-)
        uM5NULL                                 [5] NULL,

--      EXPECT [level]                          Urg(L)/Alr(L)/Resp(R)
        uM6Level                                [6] Level,

--      EXPECT CLIMB AT [time]                  Urg(L)/Alr(L)/Resp(R)
        uM7Time                                 [7] Time,

--      EXPECT CLIMB AT [position]              Urg(L)/Alr(L)/Resp(R)
        uM8Position                             [8] Position,
```

--	EXPECT DESCENT AT [time] uM9Time	Urg(L)/Alr(L)/Resp(R) [9] Time,
--	EXPECT DESCENT AT [position] uM10Position	Urg(L)/Alr(L)/Resp(R) [10] Position,
--	EXPECT CRUISE CLIMB AT [time] uM11Time	Urg(L)/Alr(L)/Resp(R) [11] Time,
--	EXPECT CRUISE CLIMB AT [position] uM12Position	Urg(L)/Alr(L)/Resp(R) [12] Position,
--	AT [time] EXPECT CLIMB TO [level] uM13TimeLevel	Urg(L)/Alr(L)/Resp(R) [13] TimeLevel,
--	AT [position] EXPECT CLIMB TO [level] uM14PositionLevel	Urg(L)/Alr(L)/Resp(R) [14] PositionLevel,
--	AT [time] EXPECT DESCENT TO [level] uM15TimeLevel	Urg(L)/Alr(L)/Resp(R) [15] TimeLevel,
--	AT [position] EXPECT DESCENT TO [level] uM16PositionLevel	Urg(L)/Alr(L)/Resp(R) [16] PositionLevel,
--	AT [time] EXPECT CRUISE CLIMB TO [level] uM17TimeLevel	Urg(L)/Alr(L)/Resp(R) [17] TimeLevel,
--	AT [position] EXPECT CRUISE CLIMB TO [level] uM18PositionLevel	Urg(L)/Alr(L)/Resp(R) [18] PositionLevel,
--	MAINTAIN [level] uM19Level	Urg(N)/Alr(M)/Resp(W/U) [19] Level,
--	CLIMB TO [level] uM20Level	Urg(N)/Alr(M)/Resp(W/U) [20] Level,
--	AT [time] CLIMB TO [level] uM21TimeLevel	Urg(N)/Alr(M)/Resp(W/U) [21] TimeLevel,
--	AT [position] CLIMB TO [level] uM22PositionLevel	Urg(N)/Alr(M)/Resp(W/U) [22] PositionLevel,
--	DESCEND TO [level] uM23Level	Urg(N)/Alr(M)/Resp(W/U) [23] Level,

```
--  AT [time] DESCEND TO [level]                    Urg(N)/Alr(M)/Resp(W/U)
    uM24TimeLevel                                   [24] TimeLevel,

--  AT [position] DESCEND TO [level]                Urg(N)/Alr(M)/Resp(W/U)
    uM25PositionLevel                               [25] PositionLevel,

--  CLIMB TO REACH [level] BY [time]                Urg(N)/Alr(M)/Resp(W/U)
    uM26LevelTime                                   [26] LevelTime,

--  CLIMB TO REACH [level] BY [position]            Urg(N)/Alr(M)/Resp(W/U)
    uM27LevelPosition                               [27] LevelPosition,

--  DESCEND TO REACH [level] BY [time]              Urg(N)/Alr(M)/Resp(W/U)
    uM28LevelTime                                   [28] LevelTime,

--  DESCEND TO REACH [level] BY [position]          Urg(N)/Alr(M)/Resp(W/U)
    uM29LevelPosition                               [29] LevelPosition,

--  MAINTAIN BLOCK [level] TO [level]               Urg(N)/Alr(M)/Resp(W/U)
    uM30LevelLevel                                  [30] LevelLevel,

--  CLIMB TO AND MAINTAIN BLOCK [level] TO [level]
--                                                  Urg(N)/Alr(M)/Resp(W/U)
    uM31LevelLevel                                  [31] LevelLevel,

--  DESCEND TO AND MAINTAIN BLOCK [level] TO [level]
--                                                  Urg(N)/Alr(M)/Resp(W/U)
    uM32LevelLevel                                  [32] LevelLevel,

--  Reserved                                        Urg(L)/Alr(L)/Resp(Y)
    uM33NULL                                        [33] NULL,

--  CRUISE CLIMB TO [level]                         Urg(N)/Alr(M)/Resp(W/U)
    uM34Level                                       [34] Level,

--  CRUISE CLIMB ABOVE [level]                      Urg(N)/Alr(M)/Resp(W/U)
    uM35Level                                       [35] Level,

--  EXPEDITE CLIMB TO [level]                       Urg(U)/Alr(M)/Resp(W/U)
    uM36Level                                       [36] Level,

--  EXPEDITE DESCENT TO [level]                     Urg(U)/Alr(M)/Resp(W/U)
    uM37Level                                       [37] Level,

--  IMMEDIATELY CLIMB TO [level]                    Urg(D)/Alr(H)/Resp(W/U)
    uM38Level                                       [38] Level,
```

```
--  IMMEDIATELY DESCEND TO [level]                  Urg(D)/Alr(H)/Resp(W/U)
    uM39Level                                       [39] Level,

--  Reserved                                        Urg(L)/Alr(L)/Resp(Y)
    uM40NULL                                        [40] NULL,

--  Reserved                                        Urg(L)/Alr(L)/Resp(W/Y)
    uM41NULL                                        [41] NULL,

--  EXPECT TO CROSS [position] AT [level]           Urg(L)/Alr(L)/Resp(R)
    uM42PositionLevel                               [42] PositionLevel,

--  EXPECT TO CROSS [position] AT OR ABOVE [level]
--                                                  Urg(L)/Alr(L)/Resp(R)
    uM43PositionLevel                               [43] PositionLevel,

--  EXPECT TO CROSS [position] AT OR BELOW [level]
--                                                  Urg(L)/Alr(L)/Resp(R)
    uM44PositionLevel                               [44] PositionLevel,

--  EXPECT TO CROSS [position] AT AND MAINTAIN [level]
--                                                  Urg(L)/Alr(L)/Resp(R)
    uM45PositionLevel                               [45] PositionLevel,

--  CROSS [position] AT [level]                     Urg(N)/Alr(M)/Resp(W/U)
    uM46PositionLevel                               [46] PositionLevel,

--  CROSS [position] AT OR ABOVE [level]            Urg(N)/Alr(M)/Resp(W/U)
    uM47PositionLevel                               [47] PositionLevel,

--  CROSS [position] AT OR BELOW [level]            Urg(N)/Alr(M)/Resp(W/U)
    uM48PositionLevel                               [48]PositionLevel,

--  CROSS [position] AT AND MAINTAIN [level]        Urg(N)/Alr(M)/Resp(W/U)
    uM49PositionLevel                               [49] PositionLevel,

--  CROSS [position] BETWEEN [level] AND [level]    Urg(N)/Alr(M)/Resp(W/U)
    uM50PositionLevelLevel                          [50] PositionLevelLevel,

--  CROSS [position] AT [time]                      Urg(N)/Alr(M)/Resp(W/U)
    uM51PositionTime                                [51] PositionTime,

--  CROSS [position] AT OR BEFORE [time]            Urg(N)/Alr(M)/Resp(W/U)
    uM52PositionTime                                [52] PositionTime,
```

```
--  CROSS [position] AT OR AFTER [time]                 Urg(N)/Alr(M)/Resp(W/U)
    uM53PositionTime                                    [53] PositionTime,

--  CROSS [position] BETWEEN [time] AND [time]          Urg(N)/Alr(M)/Resp(W/U)
    uM54PositionTimeTime                                [54] PositionTimeTime,

--  CROSS [position] AT [speed]                         Urg(N)/Alr(M)/Resp(W/U)
    uM55PositionSpeed                                   [55] PositionSpeed,

--  CROSS [position] AT OR LESS THAN [speed]            Urg(N)/Alr(M)/Resp(W/U)
    uM56PositionSpeed                                   [56] PositionSpeed,

--  CROSS [position] AT OR GREATER THAN [speed]
--                                                      Urg(N)/Alr(M)/Resp(W/U)
    uM57PositionSpeed                                   [57] PositionSpeed,

--  CROSS [position] AT [time] AT [level]               Urg(N)/Alr(M)/Resp(W/U)
    uM58PositionTimeLevel                               [58] PositionTimeLevel,

--  CROSS [position] AT OR BEFORE [time] AT [level]     Urg(N)/Alr(M)/Resp(W/U)
    uM59PositionTimeLevel                               [59] PositionTimeLevel,

--  CROSS [position] AT OR AFTER [time] AT [level]      Urg(N)/Alr(M)/Resp(W/U)
    uM60PositionTimeLevel                               [60] PositionTimeLevel,

--  CROSS [position] AT AND MAINTAIN [level] AT [speed]
--                                                      Urg(N)/Alr(M)/Resp(W/U)
    uM61PositionLevelSpeed                  [61] PositionLevelSpeed,

--  AT [time] CROSS [position] AT AND MAINTAIN [level]
--                                                      Urg(N)/Alr(M)/Resp(W/U)
    uM62TimePositionLevel                               [62] TimePositionLevel,

--  AT [time] CROSS [position] AT AND MAINTAIN [level] AT [speed]
--                                                      Urg(N)/Alr(M)/Resp(W/U)
    uM63TimePositionLevelSpeed                          [63] TimePositionLevelSpeed,

--  OFFSET [specifiedDistance] [direction] OF ROUTE     Urg(N)/Alr(M)/Resp(W/U)
    uM64DistanceSpecifiedDirection                      [64] DistanceSpecifiedDirection,

--  AT [position] OFFSET [specifiedDistance] [direction] OF ROUTE
--                                                      Urg(N)/Alr(M)/Resp(W/U)
    uM65PositionDistanceSpecifiedDirection              [65] PositionDistanceSpecifiedDirection,
```

```
--      AT [time] OFFSET [specifiedDistance] [direction] OF ROUTE
--                                                          Urg(N)/Alr(M)/Resp(W/U)
        uM66TimeDistanceSpecifiedDirection                  [66] TimeDistanceSpecifiedDirection,

--      PROCEED BACK ON ROUTE                               Urg(N)/Alr(M)/Resp(W/U)
        uM67NULL                                            [67] NULL,

--      REJOIN ROUTE BY [position]                          Urg(N)/Alr(M)/Resp(W/U)
        uM68Position                                        [68] Position,

--      REJOIN ROUTE BY [time]                              Urg(N)/Alr(M)/Resp(W/U)
        uM69Time                                            [69] Time,

--      EXPECT BACK ON ROUTE BY [position]                  Urg(L)/Alr(L)/Resp(R)
        uM70Position                                        [70] Position,

--      EXPECT BACK ON ROUTE BY [time]                      Urg(L)/Alr(L)/Resp(R)
        uM71Time                                            [71] Time,

--      RESUME OWN NAVIGATION                               Urg(N)/Alr(M)/Resp(W/U)
        uM72NULL                                            [72] NULL,

--      [DepartureClearance]                                Urg(N)/Alr(M)/Resp(W/U)
        uM73DepartureClearance                              [73] DepartureClearance,

--      PROCEED DIRECT TO [position]                        Urg(N)/Alr(M)/Resp(W/U)
        uM74Position                                        [74] Position,

--      WHEN ABLE PROCEED DIRECT TO [position] Urg(N)/Alr(M)/Resp(W/U)
        uM75Position                                        [75] Position,

--      AT [time] PROCEED DIRECT TO [position]              Urg(N)/Alr(M)/Resp(W/U)
        uM76TimePosition                                    [76] TimePosition,

--      AT [position] PROCEED DIRECT TO [position]          Urg(N)/Alr(M)/Resp(W/U)
        uM77PositionPosition                                [77] PositionPosition,

--      AT [level] PROCEED DIRECT TO [position]             Urg(N)/Alr(M)/Resp(W/U)
        uM78LevelPosition                                   [78] LevelPosition,

--      CLEARED TO [position] VIA [routeClearance]          Urg(N)/Alr(M)/Resp(W/U)
        uM79PositionRouteClearance                          [79] PositionRouteClearanceIndex,

--      CLEARED [routeClearance]                            Urg(N)/Alr(M)/Resp(W/U)
        uM80RouteClearance                                  [80] RouteClearanceIndex,
```

```
--      CLEARED [procedureName]                                  Urg(N)/Alr(M)/Resp(W/U)
        uM81ProcedureName                                        [81] ProcedureName,

--      CLEARED TO DEVIATE UP TO [specifiedDistance] [direction] OF ROUTE
--                                                               Urg(N)/Alr(M)/Resp(W/U)
        uM82DistanceSpecifiedDirection                           [82] DistanceSpecifiedDirection,

--      AT [position] CLEARED [routeClearance]                   Urg(N)/Alr(M)/Resp(W/U)
        uM83PositionRouteClearance                               [83] PositionRouteClearanceIndex,

--      AT [position] CLEARED [procedureName]                    Urg(N)/Alr(M)/Resp(W/U)
        uM84PositionProcedureName                                [84] PositionProcedureName,

--      EXPECT [routeClearance]                                  Urg(L)/Alr(L)/Resp(R)
        uM85RouteClearance                                       [85] RouteClearanceIndex,

--      AT [position] EXPECT [routeClearance]                    Urg(L)/Alr(L)/Resp(R)
        uM86PositionRouteClearance                               [86] PositionRouteClearanceIndex,

--      EXPECT DIRECT TO [position]                              Urg(L)/Alr(L)/Resp(R)
        uM87Position                                             [87] Position,

--      AT [position] EXPECT DIRECT TO [position]                Urg(L)/Alr(L)/Resp(R)
        uM88PositionPosition                                     [88]PositionPosition,

--      AT [time] EXPECT DIRECT TO [position]                    Urg(L)/Alr(L)/Resp(R)
        uM89TimePosition                                         [89] TimePosition,

--      AT [level] EXPECT DIRECT TO [position]                   Urg(L)/Alr(L)/Resp(R)
        uM90LevelPosition                                        [90] LevelPosition,

--      HOLD AT [position] MAINTAIN [level] INBOUND TRACK [degrees][direction]
--      TURNS [legtype]                                          Urg(N)/Alr(M)/Resp(W/U)
        uM91HoldClearance                                        [91] HoldClearance,

--      HOLD AT [position] AS PUBLISHED MAINTAIN [level]
--                                                               Urg(N)/Alr(M)/Resp(W/U)
        uM92PositionLevel                                        [92] PositionLevel,

--      EXPECT FURTHER CLEARANCE AT [time]                       Urg(L)/Alr(L)/Resp(R)
        uM93Time                                                 [93] Time,

--      TURN [direction] HEADING [degrees]                       Urg(N)/Alr(M)/Resp(W/U)
        uM94DirectionDegrees                                     [94] DirectionDegrees,
```

--	TURN [direction] GROUND TRACK [degrees] uM95DirectionDegrees	Urg(N)/Alr(M)/Resp(W/U) [95] DirectionDegrees,
--	CONTINUE PRESENT HEADING uM96NULL	Urg(N)/Alr(M)/Resp(W/U) [96] NULL,
--	AT [position] FLY HEADING [degrees] uM97PositionDegrees	Urg(N)/Alr(M)/Resp(W/U) [97] PositionDegrees,
--	IMMEDIATELY TURN [direction] HEADING [degrees] --	Urg(D)/Alr(H)/Resp(W/U)
	uM98DirectionDegrees	[98] DirectionDegrees,
--	EXPECT [procedureName] uM99ProcedureName	Urg(L)/Alr(L)/Resp(R) [99] ProcedureName,
--	AT [time] EXPECT [speed] uM100TimeSpeed	Urg(L)/Alr(L)/Resp(R) [100] TimeSpeed,
--	AT [position] EXPECT [speed] uM101PositionSpeed	Urg(L)/Alr(L)/Resp(R) [101] PositionSpeed,
--	AT [level] EXPECT [speed] uM102LevelSpeed	Urg(L)/Alr(L)/Resp(R) [102] LevelSpeed,
--	AT [time] EXPECT [speed] TO [speed] uM103TimeSpeedSpeed	Urg(L)/Alr(L)/Resp(R) [103] TimeSpeedSpeed,
--	AT [position] EXPECT [speed] TO [speed] uM104PositionSpeedSpeed	Urg(L)/Alr(L)/Resp(R) [104] PositionSpeedSpeed,
--	AT [level] EXPECT [speed] TO [speed] uM105LevelSpeedSpeed	Urg(L)/Alr(L)/Resp(R) [105] LevelSpeedSpeed,
--	MAINTAIN [speed] uM106Speed	Urg(N)/Alr(M)/Resp(W/U) [106] Speed,
--	MAINTAIN PRESENT SPEED uM107NULL	Urg(N)/Alr(M)/Resp(W/U) [107] NULL,
--	MAINTAIN [speed] OR GREATER uM108Speed	Urg(N)/Alr(M)/Resp(W/U) [108] Speed,
--	MAINTAIN [speed] OR LESS uM109Speed	Urg(N)/Alr(M)/Resp(W/U) [109] Speed,

--	MAINTAIN [speed] TO [speed] uM110SpeedSpeed	Urg(N)/Alr(M)/Resp(W/U) [110] SpeedSpeed,
--	INCREASE SPEED TO [speed] uM111Speed	Urg(N)/Alr(M)/Resp(W/U) [111] Speed,
--	INCREASE SPEED TO [speed] OR GREATER uM112Speed	Urg(N)/Alr(M)/Resp(W/U) [112] Speed,
--	REDUCE SPEED TO [speed] uM113Speed	Urg(N)/Alr(M)/Resp(W/U) [113] Speed,
--	REDUCE SPEED TO [speed] OR LESS uM114Speed	Urg(N)/Alr(M)/Resp(W/U) [114] Speed,
--	DO NOT EXCEED [speed] uM115Speed	Urg(N)/Alr(M)/Resp(W/U) [115] Speed,
--	RESUME NORMAL SPEED uM116NULL	Urg(N)/Alr(M)/Resp(W/U) [116] NULL,
--	CONTACT [unitname] [frequency] uM117UnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [117] UnitNameFrequency,
--	AT [position] CONTACT [unitname] [frequency] uM118PositionUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [118] PositionUnitNameFrequency,
--	AT [time] CONTACT [unitname] [frequency] uM119TimeUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [119] TimeUnitNameFrequency,
--	MONITOR [unitname] [frequency] uM120UnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [120] UnitNameFrequency,
--	AT [position] MONITOR [unitname] [frequency] uM121PositionUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [121] PositionUnitNameFrequency,
--	AT [time] MONITOR [unitname] [frequency] uM122TimeUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [122] TimeUnitNameFrequency,
--	SQUAWK [code] uM123Code	Urg(N)/Alr(M)/Resp(W/U) [123] Code,
--	STOP SQUAWK uM124NULL	Urg(N)/Alr(M)/Resp(W/U) [124] NULL,

```
--      SQUAWK MODE CHARLIE                                 Urg(N)/Alr(M)/Resp(W/U)
        uM125NULL                                           [125] NULL,

--      STOP SQUAWK MODE CHARLIE                            Urg(N)/Alr(M)/Resp(W/U)
        uM126NULL                                           [126] NULL,

--      REPORT BACK ON ROUTE                                Urg(N)/Alr(M)/Resp(W/U)
        uM127NULL                                           [127] NULL,

--      REPORT LEAVING [level]                              Urg(N)/Alr(M)/Resp(W/U)
        uM128Level                                          [128] Level,

--      REPORT MAINTAINING [level]                  Urg(N)/Alr(M)/Resp(W/U)
        uM129Level                                          [129] Level,

--      REPORT PASSING [position]                           Urg(N)/Alr(M)/Resp(W/U)
        uM130Position                                       [130] Position,

--      REPORT REMAINING FUEL AND PERSONS ON BOARD
--                                                          Urg(N)/Alr(M)/Resp(Y)
        uM131NULL                                           [131] NULL,

--      REPORT POSITION                                     Urg(N)/Alr(M)/Resp(Y)
        uM132NULL                                           [132] NULL,

--      REPORT PRESENT LEVEL                                Urg(N)/Alr(M)/Resp(Y)
        uM133NULL                                           [133] NULL,

--      REPORT [speedtype] [speedtype] [speedtype]SPEED     Urg(N)/Alr(M)/Resp(Y)
        uM134SpeedTypeSpeedTypeSpeedType                    [134] SpeedTypeSpeedTypeSpeedType,

--      CONFIRM ASSIGNED LEVEL                              Urg(N)/Alr(M)/Resp(Y)
        uM135NULL                                           [135] NULL,

--      CONFIRM ASSIGNED SPEED                              Urg(N)/Alr(M)/Resp(Y)
        uM136NULL                                           [136] NULL,

--      CONFIRM ASSIGNED ROUTE                              Urg(N)/Alr(M)/Resp(Y)
        uM137NULL                                           [137] NULL,

--      CONFIRM TIME OVER REPORTED WAYPOINT                 Urg(N)/Alr(M)/Resp(Y)
        uM138NULL                                           [138] NULL,

--      CONFIRM REPORTED WAYPOINT                           Urg(N)/Alr(M)/Resp(Y)
        uM139NULL                                           [139] NULL,
```

```
--      CONFIRM NEXT WAYPOINT                                   Urg(N)/Alr(M)/Resp(Y)
        uM140NULL                                               [140] NULL,

--      CONFIRM NEXT WAYPOINT ETA                               Urg(N)/Alr(M)/Resp(Y)
        uM141NULL                                               [141] NULL,

--      CONFIRM ENSUING WAYPOINT                                Urg(N)/Alr(M)/Resp(Y)
        uM142NULL                                               [142] NULL,

--      CONFIRM REQUEST                                         Urg(N)/Alr(M)/Resp(Y)
        uM143NULL                                               [143] NULL,

--      CONFIRM SQUAWK                                          Urg(N)/Alr(M)/Resp(Y)
        uM144NULL                                               [144] NULL,

--      REPORT HEADING                                          Urg(N)/Alr(M)/Resp(Y)
        uM145NULL                                               [145] NULL,

--      REPORT GROUND TRACK                                     Urg(N)/Alr(M)/Resp(Y)
        uM146NULL                                               [146] NULL,

--      REQUEST POSITION REPORT                                 Urg(N)/Alr(M)/Resp(Y )
        uM147NULL                                               [147] NULL,

--      WHEN CAN YOU ACCEPT [level]                             Urg(N)/Alr(M)/Resp(Y)
        uM148Level                                              [148] Level,

--      CAN YOU ACCEPT [level] AT [position]                    Urg(N)/Alr(M)/Resp(A/N)
        uM149LevelPosition                                      [149] LevelPosition,

--      CAN YOU ACCEPT [level] AT [time]                        Urg(N)/Alr(M)/Resp(A/N)
        uM150LevelTime                                          [150] LevelTime,

--      WHEN CAN YOU ACCEPT [speed]                             Urg(N)/Alr(M)/Resp(Y)
        uM151Speed                                              [151] Speed,

--      WHEN CAN YOU ACCEPT [specifiedDistance] [direction] OFFSET
--                                                              Urg(N)/Alr(M)/Resp(Y)
        uM152DistanceSpecifiedOffsetDirection                   [152] DistanceSpecifiedDirection,

--      ALTIMETER [altimeter]                                   Urg(N)/Alr(M)/Resp(R)
        uM153Altimeter                                          [153] Altimeter,

--      RADAR SERVICE TERMINATED                                Urg(N)/Alr(M)/Resp(R)
        uM154NULL                                               [154] NULL,
```

--	RADAR CONTACT [position] uM155Position	Urg(N)/Alr(M)/Resp(R) [155] Position,
--	RADAR CONTACT LOST uM156NULL	Urg(N)/Alr(M)/Resp(R) [156] NULL,
--	CHECK STUCK MICROPHONE [frequency] uM157Frequency	Urg(U)/Alr(M)/Resp(N) [157] Frequency,
--	ATIS [atiscode] uM158AtisCode	Urg(N)/Alr(M)/Resp(R) [158] ATISCode,
--	ERROR [errorInformation] uM159ErrorInformation	Urg(U)/Alr(M)/Resp(N) [159] ErrorInformation,
--	NEXT DATA AUTHORITY [facility] uM160Facility	Urg(L)/Alr(N)/Resp(N) [160] Facility,
--	END SERVICE uM161NULL	Urg(L)/Alr(N)/Resp(N) [161] NULL,
--	SERVICE UNAVAILABLE uM162NULL	Urg(L)/Alr(L)/Resp(N) [162] NULL,
--	[facilitydesignation] uM163FacilityDesignation	Urg(L)/Alr(N)/Resp(N) [163] FacilityDesignation,
--	WHEN READY uM164NULL	Urg(L)/Alr(N)/Resp(N) [164] NULL,
--	THEN uM165NULL	Urg(L)/Alr(N)/Resp(N) [165] NULL,
--	DUE TO [traffictype]TRAFFIC uM166TrafficType	Urg(L)/Alr(N)/Resp(N) [166] TrafficType,
--	DUE TO AIRSPACE RESTRICTION uM167NULL	Urg(L)/Alr(N)/Resp(N) [167] NULL,
--	DISREGARD uM168NULL	Urg(N)/Alr(M)/Resp(R) [168] NULL,
--	[freetext] uM169FreeText	Urg(N)/Alr(L)/Resp(R) [169] FreeText,

--	[freetext] uM170FreeText	Urg(D)/Alr(H)/Resp(R) [170] FreeText,
--	CLIMB AT [verticalRate] MINIMUM uM171VerticalRate	Urg(N)/Alr(M)/Resp(W/U) [171] VerticalRate,
--	CLIMB AT [verticalRate] MAXIMUM uM172VerticalRate	Urg(N)/Alr(M)/Resp(W/U) [172] VerticalRate,
--	DESCEND AT [verticalRate] MINIMUM uM173VerticalRate	Urg(N)/Alr(M)/Resp(W/U) [173] VerticalRate,
--	DESCEND AT [verticalRate] MAXIMUM uM174VerticalRate	Urg(N)/Alr(M)/Resp(W/U) [174] VerticalRate,
--	REPORT REACHING [level] uM175Level	Urg(N)/Alr(M)/Resp(W/U) [175] Level,
--	MAINTAIN OWN SEPARATION AND VMC uM176NULL	Urg(N)/Alr(M)/Resp(W/U) [176] NULL,
--	AT PILOTS DISCRETION uM177NULL	Urg(L)/Alr(L)/Resp(N) [177] NULL,
--	*Reserved* uM178NULL	Urg(L)/Alr(L)/Resp(Y) [178] NULL,
--	SQUAWK IDENT uM179NULL	Urg(N)/Alr(M)/Resp(W/U) [179] NULL,
--	REPORT REACHING BLOCK [level] TO [level] uM180LevelLevel	Urg(N)/Alr(M)/Resp(W/U) [180] LevelLevel,
--	REPORT DISTANCE [tofrom] [position] uM181ToFromPosition	Urg(N)/Alr(M)/Resp(Y) [181] ToFromPosition,
--	CONFIRM ATIS CODE uM182NULL	Urg(N)/Alr(M)/Resp(Y) [182] NULL,
--	[freetext] uM183FreeText	Urg(N)/Alr(M)/Resp(N) [183] FreeText,
--	AT [time] REPORT DISTANCE [tofrom] [position] uM184TimeToFromPosition	Urg(N)/Alr(M)/Resp(Y) [184] TimeToFromPosition,

--	AFTER PASSING [position] CLIMB TO [level] uM185PositionLevel	Urg(N)/Alr(M)/Resp(W/U) [185] PositionLevel,
--	AFTER PASSING [position] DESCEND TO [level] uM186PositionLevel	Urg(N)/Alr(M)/Resp(W/U) [186] PositionLevel,
--	[freetext] uM187FreeText	Urg(L)/Alr(N)/Resp(N) [187] FreeText,
--	AFTER PASSING [position] MAINTAIN [speed] uM188PositionSpeed	Urg(N)/Alr(M)/Resp(W/U) [188] PositionSpeed,
--	ADJUST SPEED TO [speed] uM189Speed	Urg(N)/Alr(M)/Resp(W/U) [189] Speed,
--	FLY HEADING [degrees] uM190Degrees	Urg(N)/Alr(M)/Resp(W/U) [190] Degrees,
--	ALL ATS TERMINATED uM191NULL	Urg(N)/Alr(M)/Resp(R) [191] NULL,
--	REACH [level] BY [time] uM192LevelTime	Urg(N)/Alr(M)/Resp(W/U) [192] LevelTime,
--	IDENTIFICATION LOST uM193NULL	Urg(N)/Alr(M)/Resp(R) [193] NULL,
--	[freetext] uM194FreeText	Urg(N)/Alr(L)/Resp(Y) [194] FreeText,
--	[freetext] uM195FreeText	Urg(L)/Alr(L)/Resp(R) [195] FreeText,
--	[freetext] uM196FreeText	Urg(N)/Alr(M)/Resp(W/U) [196] FreeText,
--	[freetext] uM197FreeText	Urg(U)/Alr(M)/Resp(W/U) [197] FreeText,
--	[freetext] uM198FreeText	Urg(D)/Alr(H)/Resp(W/U) [198] FreeText,
--	[freetext] uM199FreeText	Urg(N)/Alr(L)/Resp(N) [199] FreeText,

--	REPORT REACHING uM200NULL	Urg(N)/Alr(M)/Resp(R) [200] NULL,
--	*Not Used* uM201NULL	Urg(L)/Alr(L)/Resp(N) [201] NULL,
--	*Not Used* uM202NULL	Urg(L)/Alr(L)/Resp(N) [202] NULL,
--	[freetext] uM203FreeText	Urg(N)/Alr(M)/Resp(R) [203] FreeText,
--	[freetext] uM204FreeText	Urg(N)/Alr(M)/Resp(Y) [204] FreeText,
--	[freetext] uM205FreeText	Urg(N)/Alr(M)/Resp(A/N) [205] FreeText,
--	[freetext] uM206FreeText	Urg(L)/Alr(N)/Resp(Y) [206] FreeText,
--	[freetext] uM207FreeText	Urg(L)/Alr(L)/Resp(Y) [207] FreeText,
--	[freetext] uM208FreeText	Urg(L)/Alr(L)/Resp(N) [208] FreeText,
--	REACH [level] BY [position] uM209LevelPosition	Urg(N)/Alr(M)/Resp(W/U) [209] LevelPosition,
--	IDENTIFIED [position] uM210Position	Urg(N)/Alr(M)/Resp(R) [210] Position,
--	REQUEST FORWARDED uM211NULL	Urg(N)/Alr(L)/Resp(N) [211] NULL,
--	[facilitydesignation] ATIS [atiscode] CURRENT uM212FacilityDesignationATISCode	Urg(N)/Alr(M)/Resp(R) [212] FacilityDesignationATISCode,
--	[facilitydesignation] ALTIMETER [altimeter] uM213FacilityDesignationAltimeter	Urg(N)/Alr(M)/Resp(R) [213] FacilityDesignationAltimeter,
--	RVR RUNWAY [runway] [rvr] uM214RunwayRVR	Urg(N)/Alr(M)/Resp(R) [214] RunwayRVR,

--	TURN [direction][degrees] uM215 DirectionDegrees	Urg(N)/Alr(M)/Resp(W/U) [215] DirectionDegrees,
--	REQUEST FLIGHT PLAN uM216NULL	Urg(N)/Alr(M)/Resp(Y) [216] NULL,
--	REPORT ARRIVAL uM217NULL	Urg(N)/Alr(M)/Resp(Y) [217] NULL,
--	REQUEST ALREADY RECEIVED uM218NULL	Urg(L)/Alr(N)/Resp(N) [218] NULL,
--	STOP CLIMB AT [level] uM219Level	Urg(U)/Alr(M)/Resp(W/U) [219] Level,
--	STOP DESCENT AT [level] uM220Level	Urg(U)/Alr(M)/Resp(W/U) [220] Level,
--	STOP TURN HEADING [degrees] uM221Degrees	Urg(U)/Alr(M)/Resp(W/U) [221] Degrees,
--	NO SPEED RESTRICTION uM222NULL	Urg(L)/Alr(L)/Resp(R) [222] NULL,
--	REDUCE TO MINIMUM APPROACH SPEED uM223NULL	Urg(N)/Alr(M)/Resp(W/U) [223] NULL,
--	NO DELAY EXPECTED uM224NULL	Urg(N)/Alr(L)/Resp(R) [224] NULL,
--	DELAY NOT DETERMINED uM225NULL	Urg(N)/Alr(L)/Resp(R) [225] NULL,
--	EXPECTED APPROACH TIME [time] uM226Time	Urg(N)/Alr(L)/Resp(R) [226] Time,
--	LOGICAL ACKNOWLEDGMENT uM227NULL	Urg(N)/Alr(M)/Resp(N) [227] NULL,
--	REPORT ETA [position] uM228Position	Urg(L)/Alr(L)/Resp(Y) [228] Position,
--	REPORT ALTERNATE AERODROME uM229NULL	Urg(L)/Alr(L)/Resp(Y) [229] NULL,

```
--      IMMEDIATELY                                     Urg(D)/Alr(H)/Resp(N)
        uM230NULL                                       [230] NULL,

--      STATE PREFERRED LEVEL                           Urg(L)/Alr(L)/Resp(Y)
        uM231NULL                                       [231] NULL,

--      STATE-TOP-OF-DESCENT                            Urg(L)/Alr(L)/Resp(Y)
        uM232NULL                                       [232] NULL,

--      USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED
--                                                      Urg(N)/Alr(M)/Resp(N)
        uM233NULL                                       [233] NULL,

--      FLIGHT PLAN NOT HELD                            Urg(L)/Alr(L)/Resp(N)
        uM234NULL                                       [234] NULL,

--      ROGER 7500                                      Urg(U)/Alr(H)/Resp(N)
        uM235NULL                                       [235] NULL,

--      LEAVE CONTROLLED AIRSPACE                       Urg(N)/Alr(M)/Resp(W/U)
        uM236NULL                                       [236] NULL,

        ...
        }

-------------------------------------------------------------------------------
-- Downlink message element
-- -----------------------------------------------------------------------------

ATCDownlinkMsgElementId ::= CHOICE
        {
--      WILCO                                           Urg(N)/Alr(M)/Resp(N)
        dM0NULL                                         [0] NULL,

--      UNABLE                                          Urg(N)/Alr(M)/Resp(N)
        dM1NULL                                         [1] NULL,

--      STANDBY                                         Urg(N)/Alr(M)/Resp(N)
        dM2NULL                                         [2] NULL,

--      ROGER                                           Urg(N)/Alr(M)/Resp(N)
        dM3NULL                                         [3] NULL,

--      AFFIRM                                          Urg(N)/Alr(M)/Resp(N)
        dM4NULL                                         [4] NULL,
```

```
--  NEGATIVE                                                Urg(N)/Alr(M)/Resp(N)
    dM5NULL                                                 [5] NULL,

--  REQUEST [level]                                         Urg(N)/Alr(L)/Resp(Y)
    dM6Level                                                [6] Level,

--  REQUEST BLOCK [level] TO [level]                        Urg(N)/Alr(L)/Resp(Y)
    dM7LevelLevel                                           [7] LevelLevel,

--  REQUEST CRUISE CLIMB TO [level]                         Urg(N)/Alr(L)/Resp(Y)
    dM8Level                                                [8] Level,

--  REQUEST CLIMB TO [level]                                Urg(N)/Alr(L)/Resp(Y)
    dM9Level                                                [9] Level,

--  REQUEST DESCENT TO [level]                  Urg(N)/Alr(L)/Resp(Y)
    dM10Level                                               [10] Level,

--  AT [position] REQUEST CLIMB TO [level]                  Urg(N)/Alr(L)/Resp(Y)
    dM11PositionLevel                                       [11] PositionLevel,

--  AT [position] REQUEST DESCENT TO [level]                Urg(N)/Alr(L)/Resp(Y)
    dM12PositionLevel                                       [12] PositionLevel,

--  AT [time] REQUEST CLIMB TO [level]                      Urg(N)/Alr(L)/Resp(Y)
    dM13TimeLevel                                           [13] TimeLevel,

--  AT [time] REQUEST DESCENT TO [level]                    Urg(N)/Alr(L)/Resp(Y)
    dM14TimeLevel                                           [14] TimeLevel,

--  REQUEST OFFSET [specifiedDistance] [direction] OF ROUTE
--                                                          Urg(N)/Alr(L)/Resp(Y)
    dM15DistanceSpecifiedDirection                          [15] DistanceSpecifiedDirection,

--  AT [position] REQUEST OFFSET [specifiedDistance] [direction] OF ROUTE
--                                                          Urg(N)/Alr(L)/Resp(Y)
    dM16PositionDistanceSpecifiedDirection                  [16] PositionDistanceSpecifiedDirection,

--  AT [time] REQUEST OFFSET [specifiedDistance] [direction] OF ROUTE
--                                                          Urg(N)/Alr(L)/Resp(Y)
    dM17TimeDistanceSpecifiedDirection                      [17] TimeDistanceSpecifiedDirection,

--  REQUEST [speed]                                         Urg(N)/Alr(L)/Resp(Y)
    dM18Speed                                               [18] Speed,
```

```
--  REQUEST [speed] TO [speed]                          Urg(N)/Alr(L)/Resp(Y)
    dM19SpeedSpeed                                      [19] SpeedSpeed,

--  REQUEST VOICE CONTACT                               Urg(N)/Alr(L)/Resp(Y)
    dM20NULL                                            [20] NULL,

--  REQUEST VOICE CONTACT [frequency]                   Urg(N)/Alr(L)/Resp(Y)
    dM21Frequency                                       [21] Frequency,

--  REQUEST DIRECT TO [position]                        Urg(N)/Alr(L)/Resp(Y)
    dM22Position                                        [22] Position,

--  REQUEST [procedureName]                             Urg(N)/Alr(L)/Resp(Y)
    dM23ProcedureName                                   [23] ProcedureName,

--  REQUEST [routeClearance]                            Urg(N)/Alr(L)/Resp(Y)
    dM24RouteClearance                                  [24] RouteClearanceIndex,

--  REQUEST [clearanceType] CLEARANCE                   Urg(N)/Alr(L)/Resp(Y)
    dM25ClearanceType                                   [25] ClearanceType,

--  REQUEST WEATHER DEVIATION TO  [position] VIA [routeClearance]
--                                                      Urg(N)/Alr(L)/Resp(Y)
    dM26PositionRouteClearance                          [26] PositionRouteClearanceIndex,

--  REQUEST WEATHER DEVIATION UP  TO [specifiedDistance] [direction] OF ROUTE
--                                                      Urg(N)/Alr(L)/Resp(Y)
    dM27DistanceSpecifiedDirection                      [27] DistanceSpecifiedDirection,

--  LEAVING [level]                                 Urg(N)/Alr(L)/Resp(N)
    dM28Level                                           [28] Level,

--  CLIMBING TO [level]                                 Urg(N)/Alr(M)/Resp(N)
    dM29Level                                           [29]Level,

--  DESCENDING TO [level]                               Urg(N)/Alr(M)/Resp(N)
    dM30Level                                           [30] Level,

--  PASSING [position]                                  Urg(N)/Alr(M)/Resp(N)
    dM31Position                                        [31] Position,

--  PRESENT LEVEL [level]                               Urg(N)/Alr(M)/Resp(N)
    dM32Level                                           [32] Level,

--  PRESENT POSITION [position]                         Urg(N)/Alr(M)/Resp(N)
    dM33Position                                        [33] Position,
```

--	PRESENT SPEED [speed] dM34Speed	Urg(N)/Alr(M)/Resp(N) [34] Speed,
--	PRESENT HEADING [degrees] dM35Degrees	Urg(N)/Alr(M)/Resp(N) [35] Degrees,
--	PRESENT GROUND TRACK [degrees] dM36Degrees	Urg(N)/Alr(M)/Resp(N) [36] Degrees,
--	MAINTAINING [level] dM37Level	Urg(N)/Alr(M)/Resp(N) [37] Level,
--	ASSIGNED LEVEL [level] dM38Level	Urg(N)/Alr(M)/Resp(N) [38] Level,
--	ASSIGNED SPEED [speed] dM39Speed	Urg(N)/Alr(M)/Resp(N) [39] Speed,
--	ASSIGNED ROUTE [routeClearance] dM40RouteClearance	Urg(N)/Alr(M)/Resp(N) [40] RouteClearanceIndex,
--	BACK ON ROUTE dM41NULL	Urg(N)/Alr(M)/Resp(N) [41] NULL,
--	NEXT WAYPOINT [position] dM42Position	Urg(N)/Alr(M)/Resp(N) [42] Position,
--	NEXT WAYPOINT ETA [time] dM43Time	Urg(N)/Alr(M)/Resp(N) [43] Time,
--	ENSUING WAYPOINT [position] dM44Position	Urg(N)/Alr(M)/Resp(N) [44] Position,
--	REPORTED WAYPOINT [position] dM45Position	Urg(N)/Alr(M)/Resp(N) [45] Position,
--	REPORTED WAYPOINT [time] dM46Time	Urg(N)/Alr(M)/Resp(N) [46] Time,
--	SQUAWKING [code] dM47Code	Urg(N)/Alr(M)/Resp(N) [47] Code,
--	POSITION REPORT [positionreport] dM48PositionReport	Urg(N)/Alr(M)/Resp(N) [48] PositionReport,

```
--  WHEN CAN WE EXPECT [speed]                          Urg(L)/Alr(L)/Resp(Y)
    dM49Speed                                           [49] Speed,

--  WHEN CAN WE EXPECT [speed] TO [speed]               Urg(L)/Alr(L)/Resp(Y)
    dM50SpeedSpeed                                      [50] SpeedSpeed,

--  WHEN CAN WE EXPECT BACK ON ROUTE                    Urg(L)/Alr(L)/Resp(Y)
    dM51NULL                                            [51] NULL,

--  WHEN CAN WE EXPECT LOWER LEVEL                      Urg(L)/Alr(L)/Resp(Y)
    dM52NULL                                            [52] NULL,

--  WHEN CAN WE EXPECT HIGHER LEVEL                     Urg(L)/Alr(L)/Resp(Y)
    dM53NULL                                            [53] NULL,

--  WHEN CAN WE EXPECT CRUISE CLIMB TO [level]
--                                                      Urg(L)/Alr(L)/Resp(Y)
    dM54Level                                           [54] Level,

--  PAN PAN PAN                                         Urg(U)/Alr(H)/Resp(Y)
    dM55NULL                                            [55] NULL,

--  MAYDAY MAYDAY MAYDAY                                Urg(D)/Alr(H)/Resp(Y)
    dM56NULL                                            [56] NULL,

--  [remainingFuel] OF FUEL REMAINING AND [personsonboard] PERSONS ON BOARD
--                                                      Urg(U)/Alr(H)/Resp(Y)
    dM57RemainingFuelPersonsOnBoard                     [57] RemainingFuelPersonsOnBoard,

--  CANCEL EMERGENCY                                    Urg(U)/Alr(M)/Resp(Y)
    dM58NULL                                            [58] NULL,

--  DIVERTING TO [position] VIA [routeClearance]        Urg(U)/Alr(H)/Resp(Y)
    dM59PositionRouteClearance                          [59] PositionRouteClearanceIndex,

--  OFFSETTING [specifiedDistance] [direction] OF ROUTE Urg(U)/Alr(H)/Resp(Y)
    dM60DistanceSpecifiedDirection                      [60] DistanceSpecifiedDirection,

--  DESCENDING TO [level]                               Urg(U)/Alr(H)/Resp(Y)
    dM61Level                                           [61] Level,

--  ERROR [errorInformation]                            Urg(U)/Alr(L)/Resp(N)
    dM62ErrorInformation                                [62] ErrorInformation,
```

```
--  NOT CURRENT DATA AUTHORITY                  Urg(L)/Alr(L)/Resp(N)
    dM63NULL                                    [63] NULL,

--  [facilitydesignation]                       Urg(L)/Alr(L)/Resp(N)
    dM64FacilityDesignation                     [64] FacilityDesignation,

--  DUE TO WEATHER                              Urg(L)/Alr(L)/Resp(N)
    dM65NULL                                    [65] NULL,

--  DUE TO AIRCRAFT PERFORMANCE                 Urg(L)/Alr(L)/Resp(N)
    dM66NULL                                    [66] NULL,

--  [freetext]                                  Urg(N)/Alr(L)/Resp(N)
    dM67FreeText                                [67] FreeText,

--  [freetext]                                  Urg(D)/Alr(H)/Resp(Y)
    dM68FreeText                                [68] FreeText,

--  REQUEST VMC DESCENT                         Urg(N)/Alr(L)/Resp(Y)
    dM69NULL                                    [69] NULL,

--  REQUEST HEADING [degrees]                   Urg(N)/Alr(L)/Resp(Y)
    dM70Degrees                                 [70] Degrees,

--  REQUEST GROUND TRACK [degrees]              Urg(N)/Alr(L)/Resp(Y)
    dM71Degrees                                 [71] Degrees,

--  REACHING [level]                            Urg(N)/Alr(M)/Resp(N)
    dM72Level                                   [72] Level,

--  [versionnumber]                             Urg(L)/Alr(L)/Resp(N)
    dM73Versionnumber                           [73] VersionNumber,

--  REQUEST TO MAINTAIN OWN SEPARATION AND VMC
--                                              Urg(L)/Alr(L)/Resp(Y)
    dM74NULL                                    [74] NULL,

--  AT PILOTS DISCRETION                        Urg(L)/Alr(L)/Resp(N)
    dM75NULL                                    [75] NULL,

--  REACHING BLOCK [level] TO [level]           Urg(N)/Alr(N)/Resp(N)
    dM76LevelLevel                              [76] LevelLevel,

--  ASSIGNED BLOCK [level] TO [level]           Urg(N)/Alr(N)/Resp(N)
    dM77LevelLevel                              [77] LevelLevel,
```

```
--  AT [time] [distance] [tofrom] [position]                Urg(N)/Alr(N)/Resp(N)
    dM78TimeDistanceToFromPosition                          [78] TimeDistanceToFromPosition,

--  ATIS [atiscode]                                         Urg(N)/Alr(M)/Resp(N)
    dM79AtisCode                                            [79] ATISCode,

--  DEVIATING UP TO [specifiedDistance] [direction] OF ROUTE
--                                                          Urg(N)/Alr(M)/Resp(Y)
    dM80DistanceSpecifiedDirection                          [80] DistanceSpecifiedDirection,

--  WE CAN ACCEPT [level] AT [time]                         Urg(N)/Alr(L)/Resp(N)
    dM81LevelTime                                           [81] LevelTime,

--  WE CANNNOT ACCEPT [level]                               Urg(N)/Alr(L)/Resp(N)
    dM82Level                                               [82] Level,

--  WE CAN ACCEPT [speed] AT [time]                         Urg(N)/Alr(L)/Resp(N)
    dM83SpeedTime                                           [83] SpeedTime,

--  WE CANNNOT ACCEPT [speed]                               Urg(N)/Alr(L)/Resp(N)
    dM84Speed                                               [84] Speed,

--  WE CAN ACCEPT [specifiedDistance] [direction] AT [time]
--                                                          Urg(N)/Alr(L)/Resp(N)
    dM85DistanceSpecifiedDirectionTime                      [85] DistanceSpecifiedDirectionTime,

--  WE CANNNOT ACCEPT [specifiedDistance] [direction]       Urg(N)/Alr(L)/Resp(N)
    dM86DistanceSpecifiedDirection                          [86] DistanceSpecifiedDirection,

--  WHEN CAN WE EXPECT CLIMB TO [level]                     Urg(L)/Alr(L)/Resp(Y)
    dM87Level                                               [87] Level,

--  WHEN CAN WE EXPECT DESCENT TO [level]                   Urg(L)/Alr(L)/Resp(Y)
    dM88Level                                               [88] Level,

--  MONITORING [unitname] [frequency]                       Urg(L)/Alr(M)/Resp(N)
    dM89UnitnameFrequency                       [89] UnitNameFrequency,

--  [freetext]                                              Urg(N)/Alr(M)/Resp(N)
    dM90FreeText                                            [90] FreeText,

--  [freetext]                                              Urg(N)/Alr(L)/Resp(Y)
    dM91FreeText                                            [91] FreeText,
```

--	[freetext] dM92FreeText	Urg(L)/Alr(L)/Resp(Y) [92] FreeText,
--	[freetext] dM93FreeText	Urg(U)/Alr(H)/Resp(N) [93] FreeText,
--	[freetext] dM94FreeText	Urg(D)/Alr(H)/Resp(N) [94] FreeText,
--	[freetext] dM95FreeText	Urg(U)/Alr(M)/Resp(N) [95] FreeText,
--	[freetext] dM96FreeText	Urg(U)/Alr(L)/Resp(N) [96] FreeText,
--	[freetext] dM97FreeText	Urg(L)/Alr(L)/Resp(N) [97] FreeText,
--	[freetext] dM98FreeText	Urg(N)/Alr(N)/Resp(N) [98] FreeText,
--	CURRENT DATA AUTHORITY dM99NULL	Urg(L)/Alr(L)/Resp(N) [99] NULL,
--	LOGICAL ACKNOWLEDGMENT dM100NULL	Urg(N)/Alr(M)/Resp(N) [100] NULL,
--	REQUEST END OF SERVICE dM101NULL	Urg(L)/Alr(L)/Resp(Y) [101] NULL,
--	LANDING REPORT dM102NULL	Urg(N)/Alr(N)/Resp(N) [102] NULL,
--	CANCELLING IFR dM103NULL	Urg(N)/Alr(N)/Resp(Y) [103] NULL,
--	ETA[position][time] dM104PositionTime	Urg(L)/Alr(L)/Resp(N) [104] PositionTime,
--	ALTERNATE AERODROME[airport] dM105Airport	Urg(L)/Alr(L)/Resp(N) [105] Airport,
--	PREFERRED LEVEL[level] dM106Level	Urg(L)/Alr(L)/Resp(N) [106] Level,

```
--      NOT AUTHORIZED NEXT DATA AUTHORITY          Urg(L)/Alr(L)/Resp(N)
        dM107NULL                                   [107] NULL,

--      DE-ICING COMPLETE                           Urg(L)/Alr(L)/Resp(N)
        dM108NULL                                   [108] NULL,

--      TOP OF DESCENT [time]                       Urg(L)/Alr(L)/Resp(N)
        dM109Time                                   [109] Time,

--      TOP OF DESCENT [position]                   Urg(L)/Alr(L)/Resp(N)
        dM110Position                               [110] Position,

--      TOP OF DESCENT [time] [position]            Urg(L)/Alr(L)/Resp(N)
        dM111TimePosition                           [111] TimePosition,

--      SQUAWKING 7500                              Urg(U)/Alr(H)/Resp(N)
        dM112NULL                                   [112] NULL,

--      [speedType] [speedType] [speedType] SPEED [speed]   Urg(N)/Alr(L)/Resp(N)
        dM113SpeedTypeSpeedTypeSpeedTypeSpeed       [113]SpeedTypeSpeedTypeSpeedType
                                                         Speed,

        ...
        }
```

AircraftAddress ::= BIT STRING (SIZE(24))

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

Airport ::= IA5String (SIZE (4))

```
Altimeter ::=  CHOICE
        {
        altimeterEnglish        [0]     AltimeterEnglish,
        altimeterMetric         [1]     AltimeterMetric
        }
```

AltimeterEnglish ::= INTEGER (2200..3200)
-- unit = Inches Mercury, Range (22.00 .. 32.00), resolution = 0.01

AltimeterMetric ::= INTEGER (7500..12500)
-- unit = Hectopascal, Range (750.0..1250.0), resolution = 0.1

ATISCode ::= IA5String (SIZE (1))

ATSRouteDesignator ::= IA5String (SIZE (2..7))

```
ATWAlongTrackWaypoint ::= SEQUENCE
    {
    position            [0]   Position,
    aTWDistance         [1]   ATWDistance,
    speed               [2]   Speed                OPTIONAL,
    aTWLevels           [3]   ATWLevelSequence     OPTIONAL
    }

ATWLevel ::= SEQUENCE
    {
    atw             ATWLevelTolerance,
    level           Level
    }

ATWLevelSequence ::= SEQUENCE SIZE (1..2) OF ATWLevel

ATWLevelTolerance ::= ENUMERATED
    {
    at              (0),
    atorabove       (1),
    atorbelow       (2)
    }

ATWDistance ::= SEQUENCE
    {
    atwDistanceTolerance        ATWDistanceTolerance,
    distance                    Distance
    }

ATWDistanceTolerance ::= ENUMERATED
    {
    plus            (0),
    minus           (1)
    }
```

```
ClearanceType ::= ENUMERATED
    {
    noneSpecified    (0),
    approach         (1),
    departure        (2),
    further          (3),
    start-up         (4),
    pushback         (5),
    taxi             (6),
    take-off         (7),
    landing          (8),
    oceanic          (9),
    en-route         (10),
    downstream       (11),
    ...
    }

Code ::=  SEQUENCE SIZE (4) OF CodeOctalDigit

CodeOctalDigit ::= INTEGER (0..7)

ControlledTime ::= SEQUENCE
    {
    time             Time,
    timeTolerance    TimeTolerance
    }

Date ::= SEQUENCE
    {
    year             Year,
    month            Month,
    day              Day
    }

DateTimeGroup ::= SEQUENCE
    {
    date             Date,
    timehhmmss       Timehhmmss
    }

Day ::= INTEGER (1..31)
    --unit = Day, Range (1..31), resolution = 1

DegreeIncrement ::= INTEGER (1..20)
    --unit = Degree, Range (1..20), resolution = 1
```

```
Degrees ::=  CHOICE
    {
    degreesMagnetic         [0]     DegreesMagnetic,
    degreesTrue             [1]     DegreesTrue
    }

DegreesMagnetic ::= INTEGER (1..360)
    --unit = degree, Range (1..360), resolution = 1

DegreesTrue ::= INTEGER (1..360)
    --unit = degree, Range (1..360), resolution = 1

DepartureClearance ::= SEQUENCE
    {
    aircraftFlightIdentification    [0]     AircraftFlightIdentification,
    clearanceLimit                  [1]     Position,
    flightInformation               [2]     FlightInformation       OPTIONAL,
    furtherInstructions             [3]     FurtherInstructions     OPTIONAL
    }

DepartureMinimumInterval ::= INTEGER (1..150)
    --unit = Minute, Range (0.1..15.0), resolution = 0.1

Direction ::= ENUMERATED
    {
    left                    (0),
    right                   (1),
    eitherSide              (2),
    north                   (3),
    south                   (4),
    east                    (5),
    west                    (6),
    northEast               (7),
    northWest               (8),
    southEast               (9),
    southWest               (10)
    }

DirectionDegrees ::=  SEQUENCE
    {
    direction       Direction,
    degrees         Degrees
    }
```

```
Distance ::= CHOICE
        {
        distanceNm          [0]     DistanceNm,
        distanceKm          [1]     DistanceKm
        }

DistanceKm ::= INTEGER (0..8000)
--      unit = Kilometer, Range (0..2000), resolution = 0.25

DistanceNm ::= INTEGER (0..9999)
--      unit = Nautical Mile, Range (0..999.9), resolution = 0.1

DistanceSpecified ::= CHOICE
        {
        distanceSpecifiedNm         [0]     DistanceSpecifiedNm,
        distanceSpecifiedKm         [1]     DistanceSpecifiedKm
        }

DistanceSpecifiedDirection ::=  SEQUENCE
        {
        distanceSpecified           DistanceSpecified,
        direction                   Direction
        }

DistanceSpecifiedDirectionTime ::=  SEQUENCE
        {
        distanceSpecifiedDirection          DistanceSpecifiedDirection,
        time                                Time
        }

DistanceSpecifiedKm ::= INTEGER (1..500)
--      unit = Kilometer, Range (1..500), resolution = 1

DistanceSpecifiedNm ::= INTEGER (1..250)
--      unit = Nautical Mile, Range (1..250), resolution = 1

ErrorInformation ::=  ENUMERATED
        {
        unrecognizedMsgReferenceNumber              (0),
        logicalAcknowledgmentNotAccepted            (1),
        insufficientResources                       (2),
        invalidMessageElementCombination            (3),
        invalidMessageElement                       (4),
        ...
        }
```

```
Facility ::= CHOICE
        {
        noFacility              [0]     NULL,
        facilityDesignation     [1]     FacilityDesignation
        }

FacilityDesignation ::=  IA5String (SIZE (4..8))

FacilityFunction ::= ENUMERATED
        {
        center                  (0),
        approach                (1),
        tower                   (2),
        final                   (3),
        groundControl           (4),
        clearanceDelivery (5),
        departure               (6),
        control                 (7),
        radio                   (8),
        ...
        }

FacilityDesignationAltimeter ::= SEQUENCE
        {
        facilityDesignation             FacilityDesignation,
        altimeter                       Altimeter
        }

FacilityDesignationATISCode ::= SEQUENCE
        {
        facilityDesignation             FacilityDesignation,
        aTISCode                        ATISCode
        }

FacilityIdentification ::=  CHOICE
        {
        facilityDesignation     [0]     FacilityDesignation,
        facilityName            [1]     FacilityName
        }

FacilityName ::= IA5String (SIZE (3..18))

Fix ::= IA5String (SIZE (1..5))

FixName ::=  SEQUENCE
        {
```

```
        name                    [0]     Fix,
        latlon                  [1]     LatitudeLongitude       OPTIONAL
        }

FlightInformation ::= CHOICE
        {
        routeOfFlight           [0]     RouteInformation,
        levelsOfFlight          [1]     LevelsOfFlight,
        routeAndLevels          [2]     RouteAndLevels
        }

FreeText ::=  IA5String (SIZE (1..256))

Frequency ::=  CHOICE
        {
        frequencyhf             [0]     Frequencyhf,
        frequencyvhf            [1]     Frequencyvhf,
        frequencyuhf            [2]     Frequencyuhf,
        frequencysatchannel     [3]     Frequencysatchannel
        }

Frequencyhf ::= INTEGER (2850..28000)
--      unit = Kilohertz, Range (2850..28000), resolution = 1

Frequencysatchannel ::= NumericString (SIZE (12))
--      Frequencysatchannel corresponds to a 12 digit telephone number

Frequencyuhf ::= INTEGER  (9000..15999)
--      unit = Megahertz, Range (225.000..399.975), resolution = 0.025

Frequencyvhf ::= INTEGER (23600..27398)
--      unit = Megahertz, Range (118.0000..136.990), resolution = 0.005

FurtherInstructions ::= SEQUENCE
        {
        code                    [0]     Code                    OPTIONAL,
        frequencyDeparture      [1]     UnitNameFrequency       OPTIONAL,
        clearanceExpiryTime     [2]     Time                    OPTIONAL,
        airportDeparture        [3]     Airport                 OPTIONAL,
        airportDestination      [4]     Airport                 OPTIONAL,
        timeDeparture           [5]     TimeDeparture           OPTIONAL,
        runwayDeparture         [6]     Runway                  OPTIONAL,
        revisionNumber          [7]     RevisionNumber          OPTIONAL,
        aTISCode                [8]     ATISCode                OPTIONAL
        }
```

```
Holdatwaypoint ::= SEQUENCE
    {
    position                    [0]  Position,
    holdatwaypointspeedlow      [1]  Speed          OPTIONAL,
    aTWlevel            [2]  ATWLevel       OPTIONAL,
    holdatwaypointspeedhigh     [3]  Speed          OPTIONAL,
    direction                   [4]  Direction      OPTIONAL,
    degrees                     [5]  Degrees        OPTIONAL,
    eFCtime                     [6]  Time           OPTIONAL,
    legtype                     [7]  LegType        OPTIONAL
    }

HoldClearance ::=  SEQUENCE
    {
    position            [0]  Position,
    level               [1]  Level,
    degrees             [2]  Degrees,
    direction           [3]  Direction,
    legType             [4]  LegType        OPTIONAL
    }

Humidity ::= INTEGER (0..100)
--  unit = Percent humidity, Range (0..100), resolution = 1

InterceptCourseFrom ::= SEQUENCE
    {
    fromSelection       InterceptCourseFromSelection,
    degrees             Degrees
    }

InterceptCourseFromSelection ::= CHOICE
    {
    publishedIdentifier         [0]  PublishedIdentifier,
    latitudeLongitude           [1]  LatitudeLongitude,
    placeBearingPlaceBearing    [2]  PlaceBearingPlaceBearing,
    placeBearingDistance        [3]  PlaceBearingDistance
    }

Icing ::= ENUMERATED
    {
    trace       (0),
    light       (1),
    moderate    (2),
    severe      (3)
    }
```

```
Latitude ::= SEQUENCE
    {
    latitudeType                LatitudeType,
    latitudeDirection           LatitudeDirection
    }

LatitudeDegrees  ::= INTEGER (0..90000)
    -- unit = Degree, Range (0..90), resolution = 0.001

LatitudeDegreesMinutes ::= SEQUENCE
    {
    latitudeWholeDegrees        LatitudeWholeDegrees,
    minutesLatLon               MinutesLatLon
    }

LatitudeDegreesMinutesSeconds ::= SEQUENCE
    {
    latitudeWholeDegrees        LatitudeWholeDegrees,
    latlonWholeMinutes          LatLonWholeMinutes,
    secondsLatLon               SecondsLatLon
    }

LatitudeDirection ::= ENUMERATED
    {
    north           (0),
    south           (1)
    }

LatitudeWholeDegrees ::= INTEGER (0..89)
    -- unit = Degree, Range (0..89), resolution = 1

LatitudeLongitude ::= SEQUENCE
    {
    latitude        [0]     Latitude        OPTIONAL,
    longitude       [1]     Longitude       OPTIONAL
    }

LatitudeReportingPoints ::= SEQUENCE
    {
    latitudeDirection       LatitudeDirection,
    latitudeDegrees         LatitudeDegrees
    }

LatitudeType ::= CHOICE
    {
    latitudeDegrees             [0]     LatitudeDegrees,
```

```
    latitudeDegreesMinutes      [1]     LatitudeDegreesMinutes,
    latitudeDMS                 [2]     LatitudeDegreesMinutesSeconds
    }

LatLonWholeMinutes ::= INTEGER (0..59)
    -- unit = Minute, Range (0..59), resolution = 1

LatLonReportingPoints ::= CHOICE
    {
    latitudeReportingPoints     [0]     LatitudeReportingPoints,
    longitudeReportingPoints    [1]     LongitudeReportingPoints
    }

LegDistance ::= CHOICE
    {
    legDistanceEnglish          [0]     LegDistanceEnglish,
    legDistanceMetric           [1]     LegDistanceMetric
    }

LegDistanceEnglish ::= INTEGER (0..50)
    -- unit = Nautical Mile, Range (0..50), resolution = 1

LegDistanceMetric ::= INTEGER (1..128)
    -- unit = Kilometer, Range (1..128), resolution = 1

LegTime ::= INTEGER (0..10)
    --unit = Minute, Range (0..10), resolution = 1

LegType ::= CHOICE
    {
    legDistance     [0]     LegDistance,
    legTime         [1]     LegTime
    }

Level ::= CHOICE
    {
    singleLevel     [0]     LevelType,
    blockLevel      [1]     SEQUENCE SIZE (2) OF LevelType
    }

LevelFeet ::=  INTEGER (-60..7000)
    --unit = Feet, Range (-600..70000), resolution = 10

LevelFlightLevel ::= INTEGER (30..700)
    --unit = Level (100 Feet), Range (030..700), resolution = 1
```

LevelFlightLevelMetric ::= INTEGER (100..2500)
--unit = Level (10 Meters), Range (100..2500), resolution = 1

LevelLevel ::= SEQUENCE SIZE (2) OF Level

LevelMeters ::= INTEGER (-30..25000)
--unit = Meter, Range (-30..25000), resolution = 1

LevelPosition ::= SEQUENCE
{
level Level,
position Position
}

LevelProcedureName ::= SEQUENCE
{
level Level,
procedureName ProcedureName
}

LevelsOfFlight ::= CHOICE
{
level [0] Level,
procedureName [1] ProcedureName,
levelProcedureName [2] LevelProcedureName
}

LevelSpeed ::= SEQUENCE
{
level Level,
speed Speed
}

LevelSpeedSpeed ::= SEQUENCE
{
level Level,
speeds SpeedSpeed
}

LevelTime ::= SEQUENCE
{
level Level,
time Time
}

```
LevelType ::=  CHOICE
        {
        levelFeet                [0]    LevelFeet,
        levelMeters              [1]    LevelMeters,
        levelFlightLevel         [2]    LevelFlightLevel,
        levelFlightLevelMetric   [3]    LevelFlightLevelMetric
        }

Longitude ::= SEQUENCE
        {
        longitudeType                   LongitudeType,
        longitudeDirection              LongitudeDirection
        }

LongitudeDegrees ::= INTEGER (0..180000)
        --unit = Degree, Range (0..180), resolution = 0.001

LongitudeDegreesMinutes ::= SEQUENCE
        {
        longitudeWholeDegrees           LongitudeWholeDegrees,
        minutesLatLon                   MinutesLatLon
        }

LongitudeDegreesMinutesSeconds ::= SEQUENCE
        {
        longitudeWholeDegrees           LongitudeWholeDegrees,
        latonWholeMinutes               LatLonWholeMinutes,
        secondsLatLon                   SecondsLatLon
        }

LongitudeDirection ::= ENUMERATED
        {
        east            (0),
        west            (1)
        }

LongitudeWholeDegrees ::= INTEGER (0..179)
        -- unit = Degree, Range (0..179), resolution = 1

LongitudeReportingPoints ::= SEQUENCE
        {
        longitudeDirection              LongitudeDirection,
        longitudeDegrees                LongitudeDegrees
        }

LongitudeType ::= CHOICE
```

```
    {
    longitudeDegrees          [0]   LontitudeDegrees,
    longitudeDegreesMinutes   [1]   LongitudeDegreesMinutes,
    longitudeDMS              [2]   LongitudeDegreesMinutesSeconds
    }

MinutesLatLon ::= INTEGER (0..5999)
    --unit = Minute, Range (0..59.99), resolution = 0.01

Month ::= INTEGER (1..12)
    --unit = 1 Month, Range (1..12), resolution = 1

Navaid ::= SEQUENCE
    {
    name                      [0]   NavaidName,
    latlon                    [1]   LatitudeLongitude          OPTIONAL
    }

NavaidName ::= IA5String (SIZE (1..4))

PersonsOnBoard ::= INTEGER (1..1024)

PlaceBearing ::= SEQUENCE
    {
    publishedIdentifier             PublishedIdentifier,
    degrees                         Degrees
    }

PlaceBearingDistance ::= SEQUENCE
    {
    publishedIdentifier             PublishedIdentifier,
    degrees                         Degrees,
    distance                        Distance
    }

PlaceBearingPlaceBearing ::= SEQUENCE SIZE (2) OF PlaceBearing

Position ::= CHOICE
    {
    fixName                   [0]   FixName,
    navaid                    [1]   Navaid,
    airport                   [2]   Airport,
    latitudeLongitude         [3]   LatitudeLongitude,
    placeBearingDistance      [4]   PlaceBearingDistance
    }
```

```
PositionDegrees ::= SEQUENCE
    {
    position                    Position,
    degrees                     Degrees
    }

PositionDistanceSpecifiedDirection ::= SEQUENCE
    {
    position                    Position,
    distanceSpecifiedDirection  DistanceSpecifiedDirection
    }

PositionLevel ::= SEQUENCE
    {
    position        Position,
    level           Level
    }

PositionLevelLevel ::= SEQUENCE
    {
    position        Position,
    levels          LevelLevel
    }

PositionLevelSpeed ::= SEQUENCE
    {
    positionlevel   PositionLevel,
    speed           Speed
    }

PositionPosition ::=  SEQUENCE SIZE (2) OF Position

PositionProcedureName ::= SEQUENCE
    {
    position            Position,
    procedureName       ProcedureName
    }
```

```
PositionReport ::= SEQUENCE
    {
    positioncurrent             [0]   Position,
    timeatpositioncurrent       [1]   Time,
    level                       [2]   Level,
    fixnext                     [3]   Position          OPTIONAL,
    timeetaatfixnext            [4]   Time              OPTIONAL,
    fixnextplusone              [5]   Position          OPTIONAL,
    timeetaatdestination        [6]   Time              OPTIONAL,
    remainingFuel               [7]   RemainingFuel     OPTIONAL,
    temperature                 [8]   Temperature       OPTIONAL,
    winds                       [9]   Winds             OPTIONAL,
    turbulence                  [10]  Turbulence        OPTIONAL,
    icing                       [11]  Icing             OPTIONAL,
    speed                       [12]  Speed             OPTIONAL,
    speedground                 [13]  SpeedGround       OPTIONAL,
    verticalChange              [14]  VerticalChange    OPTIONAL,
    trackAngle                  [15]  Degrees           OPTIONAL,
    heading                     [16]  Degrees           OPTIONAL,
    distance                    [17]  Distance          OPTIONAL,
    humidity                    [18]  Humidity          OPTIONAL,
    reportedWaypointPosition [19]  Position          OPTIONAL,
    reportedWaypointTime        [20]  Time              OPTIONAL,
    reportedWaypointLevel       [21]  Level             OPTIONAL
    }

PositionRouteClearance ::= SEQUENCE
    {
    position                Position,
    routeClearance          RouteClearanceIndex
    }

PositionSpeed ::= SEQUENCE
    {
    position                Position,
    speed                   Speed
    }

PositionSpeedSpeed ::= SEQUENCE
    {
    position                Position,
    speeds                  SpeedSpeed
    }
```

```
PositionTime ::= SEQUENCE
    {
    position            Position,
    time                Time
    }

PositionTimeLevel ::= SEQUENCE
    {
    positionTime        PositionTime,
    level               Level
    }

PositionTimeTime ::= SEQUENCE
    {
    position            Position,
    times               TimeTime
    }

PositionUnitNameFrequency ::= SEQUENCE
    {
    position            Position,
    unitname            UnitName,
    frequency       Frequency
    }

Procedure ::= IA5String (SIZE (1..20))

ProcedureName ::= SEQUENCE
    {
    type            [0]     ProcedureType,
    procedure       [1]     Procedure,
    transition      [2]     ProcedureTransition         OPTIONAL
    }

ProcedureTransition ::= IA5String (SIZE (1..5))

ProcedureType ::= ENUMERATED
    {
    arrival             (0),
    approach            (1),
    departure           (2)
    }
```

```
PublishedIdentifier ::= CHOICE
    {
    fixName             [0]   FixName,
    navaid              [1]   Navaid
    }

RemainingFuel ::= Time

RemainingFuelPersonsOnBoard ::=  SEQUENCE
    {
    remainingFuel           RemainingFuel,
    personsOnBoard          PersonsOnBoard
    }

ReportingPoints ::= SEQUENCE
    {
    latLonReportingPoints   [0]   LatLonReportingPoints,
    degreeIncrement         [1]   DegreeIncrement                 OPTIONAL
    }

RevisionNumber ::= INTEGER (1..16)

RouteAndLevels ::= SEQUENCE
    {
    routeOfFlight           RouteInformation,
    levelsOfFlight          LevelsOfFlight
    }

RouteClearance ::= SEQUENCE
    {
    airportDeparture            [0]   Airport                     OPTIONAL,
    airportDestination          [1]   Airport                     OPTIONAL,
    runwayDeparture             [2]   Runway                      OPTIONAL,
    procedureDeparture          [3]   ProcedureName               OPTIONAL,
    runwayArrival               [4]   Runway                      OPTIONAL,
    procedureApproach           [5]   ProcedureName               OPTIONAL,
    procedureArrival            [6]   ProcedureName               OPTIONAL,
    routeInformations           [7]   SEQUENCE SIZE (1..128)
                                      OF RouteInformation         OPTIONAL,
    routeInformationAdditional  [8]   RouteInformationAdditional  OPTIONAL
    }
```

```
RouteClearanceIndex ::= INTEGER (1..2)
    -- RouteClearanceIndex identifies the position of the RouteClearance data
    -- in the ASN.1 type for
            -- ATC UplinkMessage, constrained Data, routeClearance Data
            -- ATC DownlinkMessage, constrained Data, routeClearance Data

RouteInformation ::= CHOICE
    {
    publishedIdentifier         [0]  PublishedIdentifier,
    latitudeLongitude           [1]  LatitudeLongitude,
    placeBearingPlaceBearing    [2]  PlaceBearingPlaceBearing,
    placeBearingDistance        [3]  PlaceBearingDistance,
    aTSRouteDesignator          [4]  ATSRouteDesignator
    }


RouteInformationAdditional ::= SEQUENCE
    {
    aTWAlongTrackWaypoints      [0]  SEQUENCE SIZE (1..8) OF ATWAlongTrackWaypoint
                                     OPTIONAL,
    reportingpoints             [1]  ReportingPoints
                                     OPTIONAL,
    interceptCourseFroms        [2]  SEQUENCE SIZE (1..4) OF InterceptCourseFrom
                                     OPTIONAL,
    holdAtWaypoints             [3]  SEQUENCE SIZE (1..8) OF Holdatwaypoint
                                     OPTIONAL,
    waypointSpeedLevels         [4]  SEQUENCE SIZE (1..32) OF WaypointSpeedLevel
                                     OPTIONAL,
    rTARequiredTimeArrivals     [5]  SEQUENCE SIZE (1..32) OF
                                     RTARequiredTimeArrival
                                     OPTIONAL
    }

RTARequiredTimeArrival ::= SEQUENCE
    {
    position                    [0]  Position,
    rTATime                     [1]  RTATime,
    rTATolerance                [2]  RTATolerance          OPTIONAL
    }

RTATime ::= SEQUENCE
    {
    time                        Time,
    timeTolerance               TimeTolerance
    }
```

```
RTATolerance ::= INTEGER (1..150)
    --unit= Minute, Range (0.1..15.0), resolution = 0.1

Runway ::= SEQUENCE
    {
    direction               RunwayDirection,
    configuration           RunwayConfiguration
    }

RunwayDirection ::= INTEGER (1..36)

RunwayConfiguration ::= ENUMERATED
    {
    left            (0),
    right           (1),
    center          (2),
    none            (3)
    }

RunwayRVR ::= SEQUENCE
    {
    runway                  Runway,
    rVR                     RVR
    }

RVR ::= CHOICE
    {
    rVRFeet         [0]     RVRFeet,
    rVRMeters       [1]     RVRMeters
    }

RVRFeet ::= INTEGER (0..6100)
    -- unit = Feet, Range (0..6100), resolution = 1

RVRMeters ::= INTEGER (0..1500)
    -- unit = Meters (0..1500), resolution = 1

SecondsLatLon ::= INTEGER (0..59)
    --unit = Second, Range (0.. 59), resolution = 1
```

```
Speed ::=  CHOICE
        {
        speedIndicated              [0]     SpeedIndicated,
        speedIndicatedMetric        [1]     SpeedIndicatedMetric,
        speedTrue                   [2]     SpeedTrue,
        speedTrueMetric             [3]     SpeedTrueMetric,
        speedGround                 [4]     SpeedGround,
        speedGroundMetric           [5]     SpeedGroundMetric,
        speedMach                   [6]     SpeedMach

        }

SpeedIndicated ::= INTEGER (0..400)
        -- unit = Knots, Range (0..400), resolution = 1

SpeedIndicatedMetric ::= INTEGER (0..800)
        -- unit = Kilometers/Hour, Range (0..800), resolution = 1

SpeedGround ::= INTEGER (-50..2000)
        -- unit = Knots, Range (-50..2000), resolution = 1

SpeedGroundMetric ::= INTEGER (-100..4000)
        -- unit = Kilometers/Hour, Range (-100..4000), resolution = 1

SpeedMach ::= INTEGER (500..4000)
        -- unit = Mach Range (0.5 to 4.0), resolution = 0.001

SpeedSpeed ::=  SEQUENCE SIZE (2) OF Speed

SpeedTime ::= SEQUENCE
        {
        speed                   Speed,
        time                    Time
        }

SpeedTrue ::= INTEGER (0..2000)
        -- unit = Knots, Range (0..2000), resolution = 1

SpeedTrueMetric ::= INTEGER (0..4000)
        -- unit = Kilometers/Hour, Range (0..4000), resolution = 1
```

```
SpeedType ::= ENUMERATED
    {
    noneSpecified    (0),
    indicated        (1),
    true             (2),
    ground           (3),
    mach             (4),
    approach         (5),
    cruise           (6),
    minimum          (7),
    maximum          (8),
    ...
    }

SpeedTypeSpeedTypeSpeedType ::= SEQUENCE SIZE (3) OF SpeedType

SpeedTypeSpeedTypeSpeedTypeSpeed ::= SEQUENCE
    {
    speedTypes       SpeedTypeSpeedTypeSpeedType,
    speed            Speed
    }

Temperature ::= INTEGER (-100..100)
    -- unit = Degree Celsius, Range (-100..100), resolution = 1

Time ::= SEQUENCE
    {
    hours            TimeHours,
    minutes          TimeMinutes
    }

TimeLevel ::= SEQUENCE
    {
    time             Time,
    level            Level
    }

TimeDeparture ::= SEQUENCE
    {
    timeDepartureAllocated              [0]  Time                       OPTIONAL,
    timeDepartureControlled             [1]  ControlledTime             OPTIONAL,
    timeDepartureClearanceExpected      [2]  Time                       OPTIONAL,
    departureMinimumInterval            [3]  DepartureMinimumInterval   OPTIONAL
    }
```

```
TimeDistanceSpecifiedDirection ::= SEQUENCE
        {
        time                            Time,
        distanceSpecifiedDirection      DistanceSpecifiedDirection
        }

TimeDistanceToFromPosition ::=  SEQUENCE
        {
        time            Time,
        distance        Distance,
        tofrom          ToFrom,
        position        Position
        }

Timehhmmss ::= SEQUENCE
        {
        hoursminutes    Time,
        seconds         TimeSeconds
        }

TimeHours ::= INTEGER (0..23)
        -- unit = Hour, Range (0..23), resolution = 1

TimeUnitNameFrequency ::=  SEQUENCE
        {
        time            Time,
        unitName        UnitName,
        frequency       Frequency
        }

TimeMinutes ::= INTEGER (0..59)
        -- unit = Minute, Range (0..59), resolution = 1

TimePosition ::=  SEQUENCE
        {
        time            Time,
        position        Position
        }

TimePositionLevel ::=    SEQUENCE
        {
        timeposition    TimePosition,
        level           Level
        }
```

```
TimePositionLevelSpeed ::= SEQUENCE
    {
    timeposition        TimePosition,
    levelspeed          LevelSpeed
    }

TimeSeconds ::= INTEGER (0..59)
    -- unit = Second, Range (0..59), resolution = 1

TimeSpeed ::=  SEQUENCE
    {
    time                Time,
    speed               Speed
    }

TimeSpeedSpeed ::=  SEQUENCE
    {
    time                Time,
    speedspeed          SpeedSpeed
    }

TimeTime ::= SEQUENCE SIZE (2) OF Time

TimeToFromPosition ::= SEQUENCE
    {
    time                Time,
    tofrom              ToFrom,
    position            Position
    }

TimeTolerance ::= ENUMERATED
    {
    at          (0),
    atorafter   (1),
    atorbefore  (2)
    }

ToFrom ::= ENUMERATED
    {
    to          (0),
    from        (1)
    }
```

```
ToFromPosition ::=  SEQUENCE
        {
        toFrom          ToFrom,
        position        Position
        }

TrafficType ::= ENUMERATED
        {
        noneSpecified           (0),
        oppositeDirection (1),
        sameDirection           (2),
        converging              (3),
        crossing                (4),
        diverging               (5),
        ...
        }

Turbulence ::= ENUMERATED
        {
        light           (0),
        moderate        (1),
        severe          (2)
        }

UnitName ::=    SEQUENCE
        {
        facilityDesignation     [0]     FacilityDesignation,
        facilityName            [1]     FacilityName            OPTIONAL,
        facilityFunction        [2]     FacilityFunction
        }

UnitNameFrequency ::=  SEQUENCE
        {
        unitName        UnitName,
        frequency       Frequency
        }

VersionNumber ::=  INTEGER (0..15)

VerticalChange ::= SEQUENCE
        {
        direction               VerticalDirection,
        rate                    VerticalRate
        }

VerticalDirection ::= ENUMERATED
```

```
    {
    up          (0),
    down        (1)
    }

VerticalRate ::=  CHOICE
    {
    verticalRateEnglish             [0]     VerticalRateEnglish,
    verticalRateMetric              [1]     VerticalRateMetric
    }

VerticalRateEnglish ::= INTEGER (0..3000)
    -- unit = Feet/Minute, Range (0..30000), resolution = 10

VerticalRateMetric ::= INTEGER (0..1000)
    -- unit = Meters/Minute, Range (0..10000), resolution = 10

WaypointSpeedLevel ::= SEQUENCE
    {
    position                [0]     Position,
    speed                   [1]     Speed                   OPTIONAL,
    aTWLevels               [2]     ATWLevelSequence        OPTIONAL
    }

WindDirection ::= INTEGER (1..360)
    -- unit = Degree, Range (1..360), resolution = 1

Winds ::= SEQUENCE
    {
    direction               WindDirection,
    speed                   WindSpeed
    }

WindSpeed ::= CHOICE
    {
    windSpeedEnglish        [0]     WindSpeedEnglish,
    windSpeedMetric         [1]     WindSpeedMetric
    }

WindSpeedEnglish ::= INTEGER (0..255)
    -- unit = Knot, Range (0..255), resolution = 1

WindSpeedMetric ::= INTEGER (0..511)
    -- unit = Kilometer/Hour, Range (0..511), resolution = 1
```

```
Year ::= INTEGER (1996..2095)
        -- unit = Year, Range (1996..2095), resolution = 1


END
```

2.3.5 PROTOCOL DEFINITION

2.3.5.1 Sequence Rules

2.3.5.1.1 With the exception of abort primitives, only the sequence of primitives illustrated in figures 2.3.5-1 to 2.3.5-18 shall be permitted.

Note 1.— The following figures define the valid sequences of primitives that are possible to be invoked during the operation of the CPDLC application. It shows the relationship in time between the service request and the resulting indication, and if applicable, the subsequent response and resulting confirmation.

Note 2.— Abort primitives may interrupt and terminate any of the normal message sequences outlined below.

Note 3.— Primitives are processed in the order received. See 4.4.3.

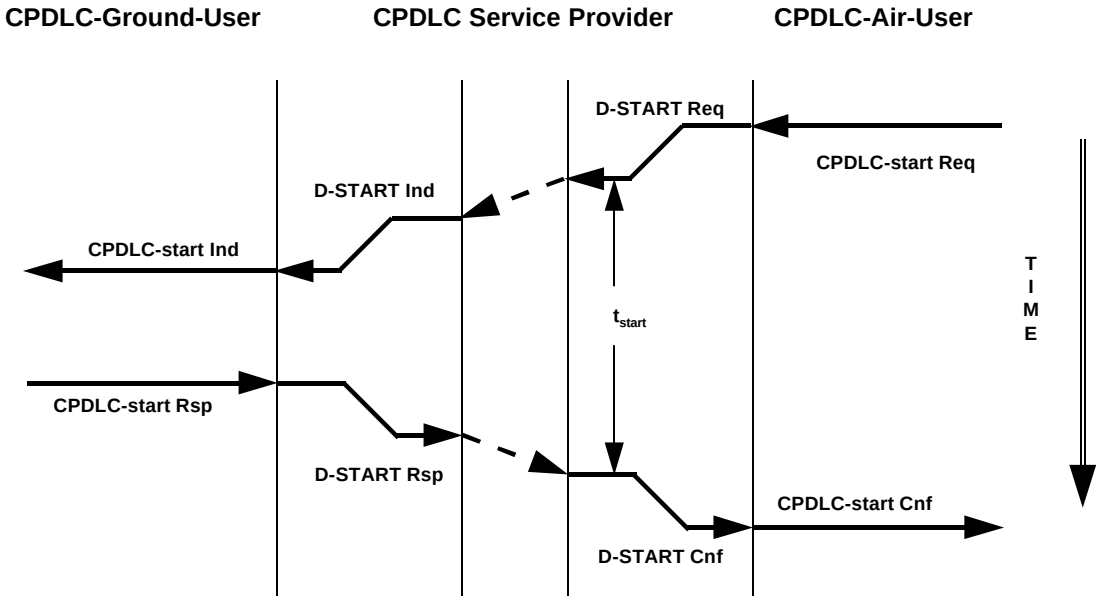

Figure 2.3.5-1. Sequence Diagram for CPDLC-start Service/Air Initiated

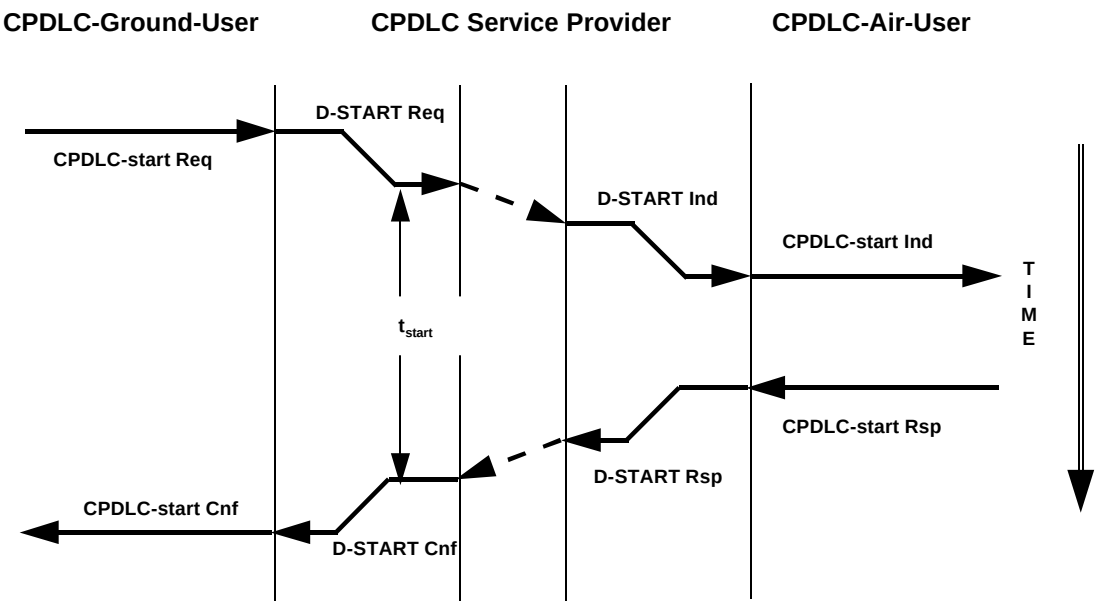

Figure 2.3.5-2. Sequence Diagram for CPDLC-start Service/Ground Initiated

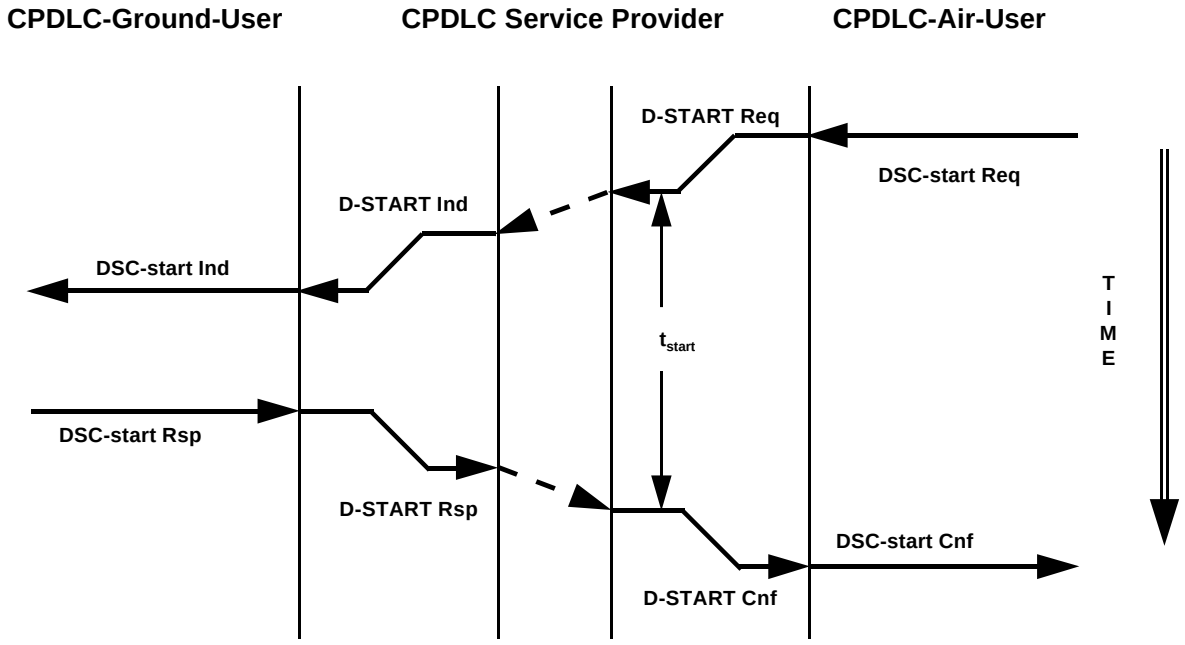

Figure 2.3.5-3. Sequence Diagram for DSC-start Service

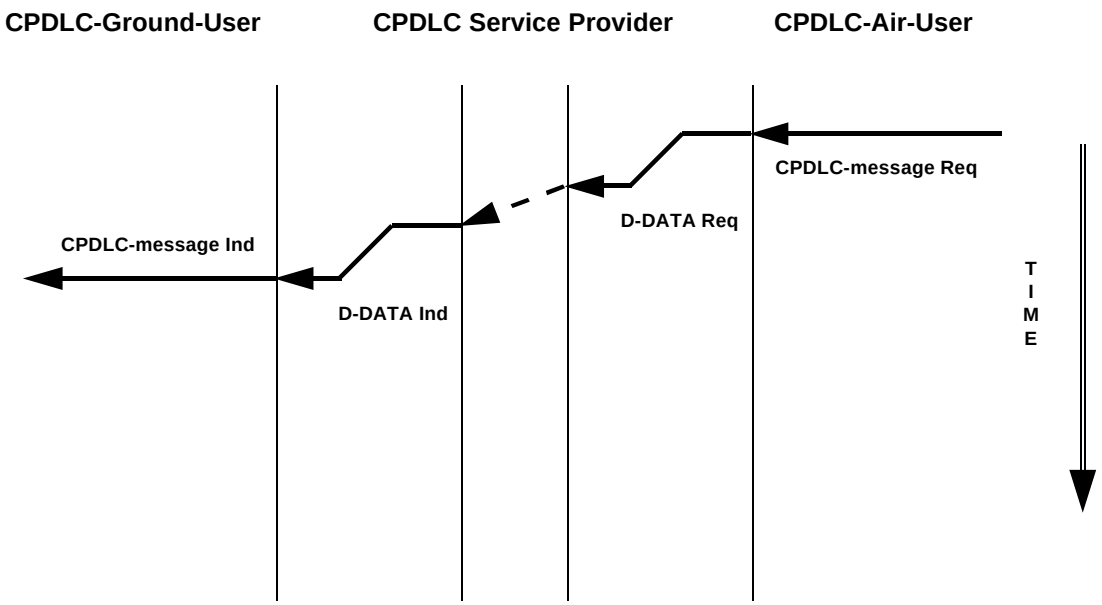

Figure 2.3.5-4. Sequence Diagram for CPDLC-message Service/Air Initiated

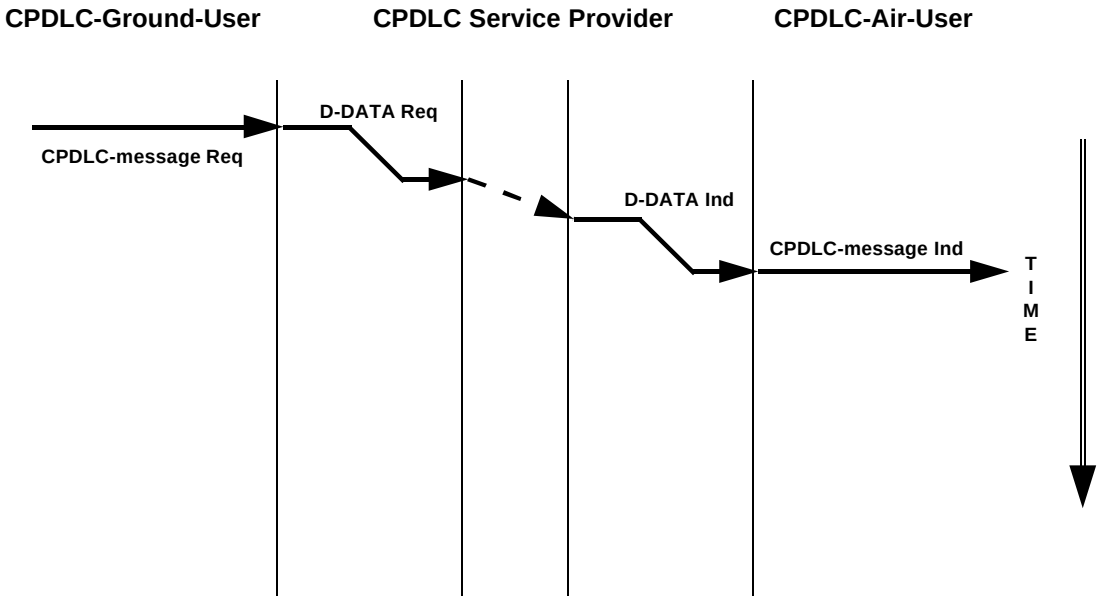

Figure 2.3.5-5. Sequence Diagram for CPDLC-message Service/Ground Initiated

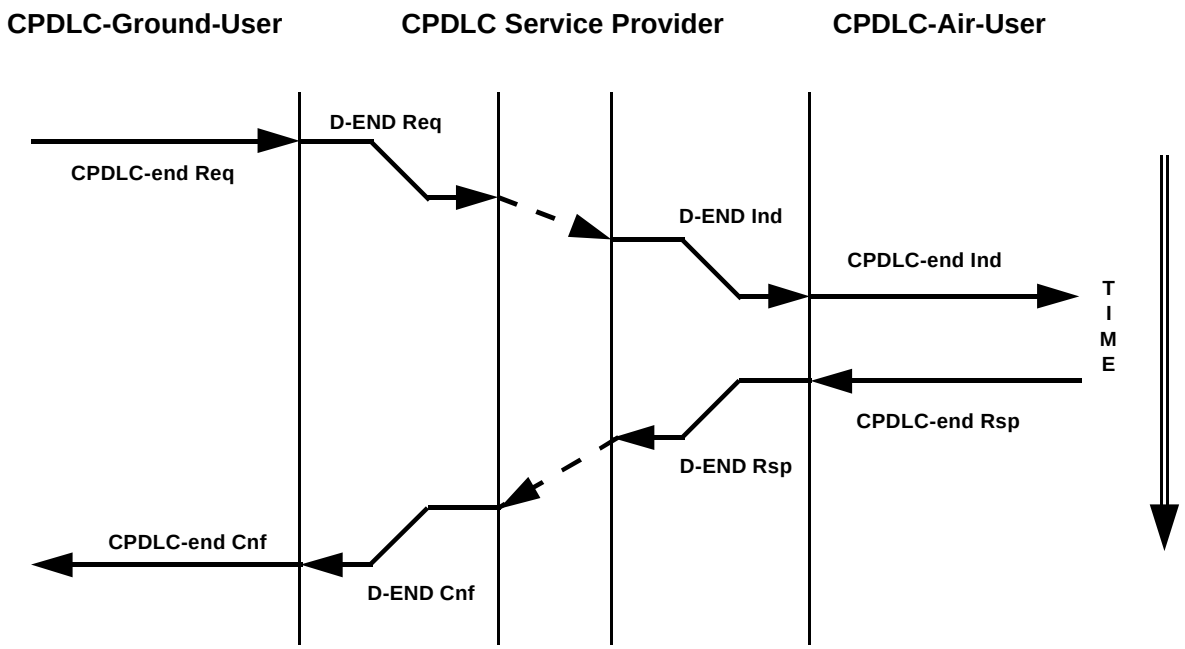

Figure 2.3.5-6. Sequence Diagram for CPDLC-end Service

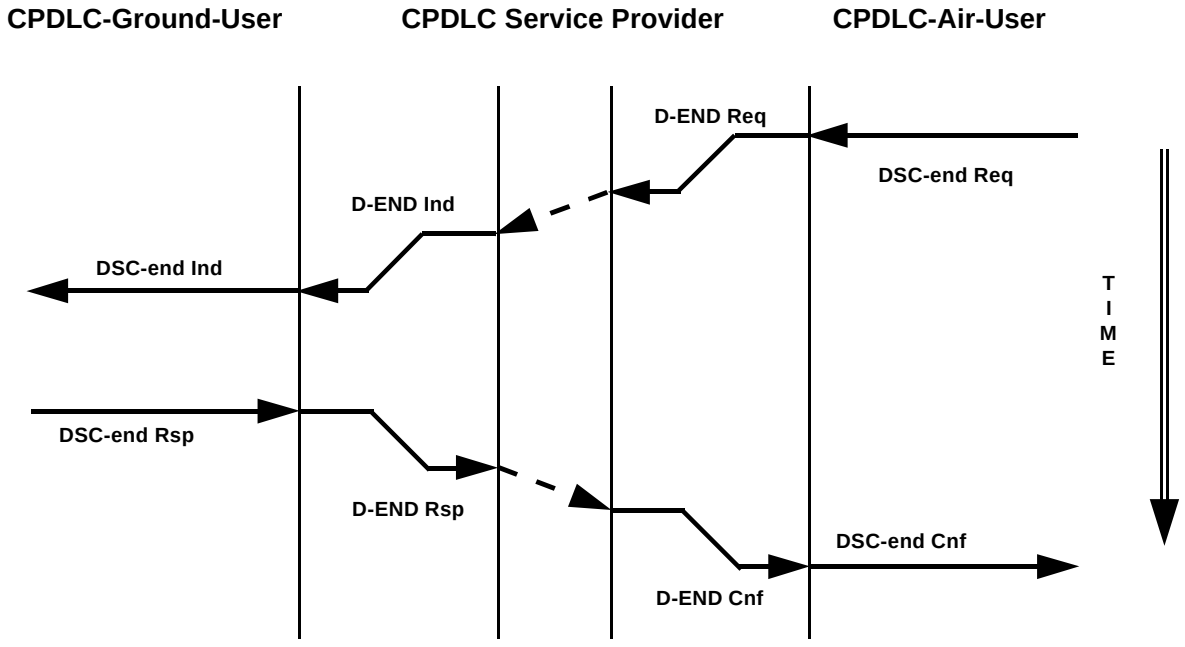

Figure 2.3.5-7. Sequence Diagram for DSC-end Service

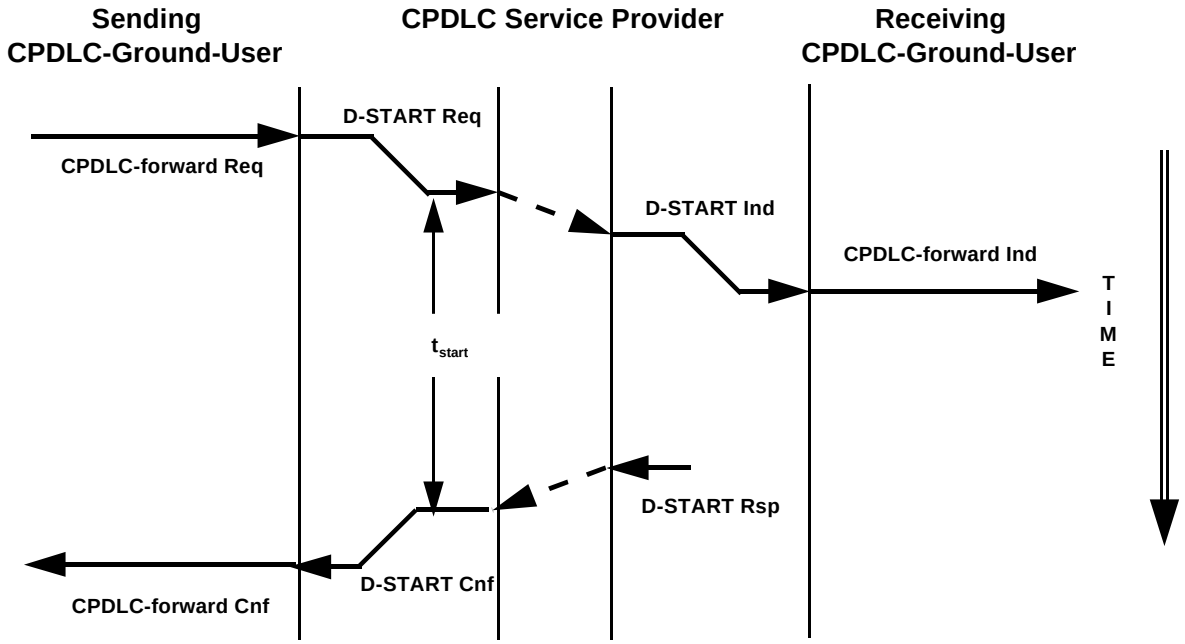

Figure 2.3.5-8. Sequence Diagram for CPDLC-forward Service/Ground Forwarding Supported, ASE Version Numbers the Same

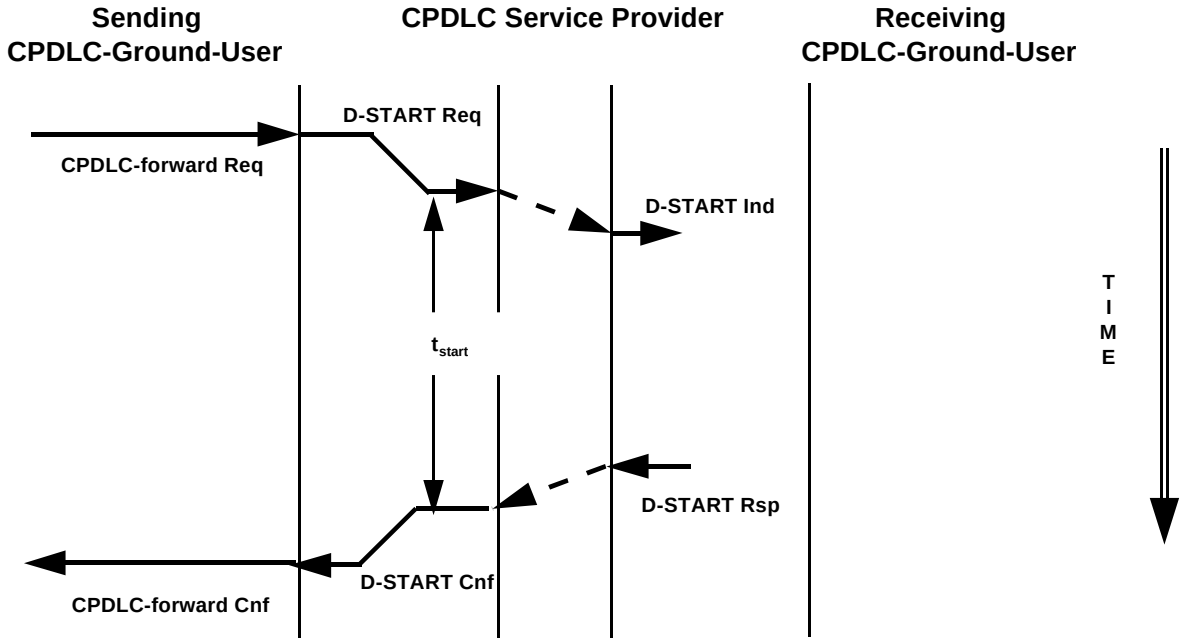

Figure 2.3.5-9. Sequence Diagram for CPDLC-forward Service/Ground Forwarding Not Supported, or Ground Forwarding Supported and ASE Version Numbers Not the Same

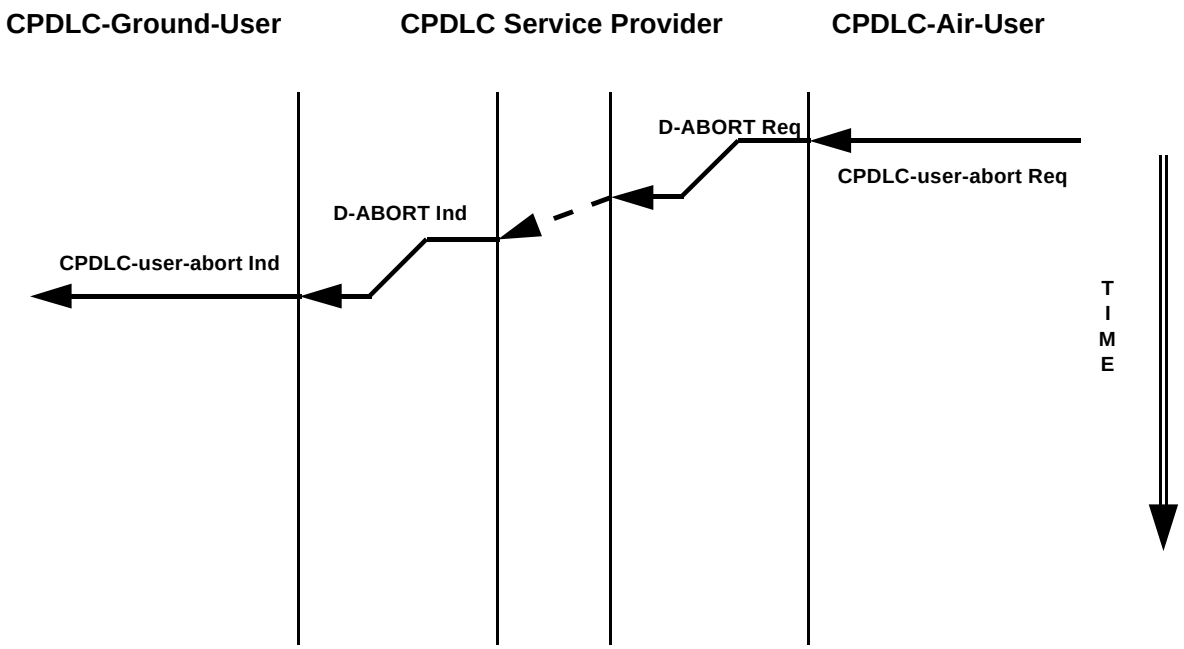

Figure 2.3.5-10. Sequence Diagram for CPDLC-user-abort Service/CPDLC-Air-User Initiated

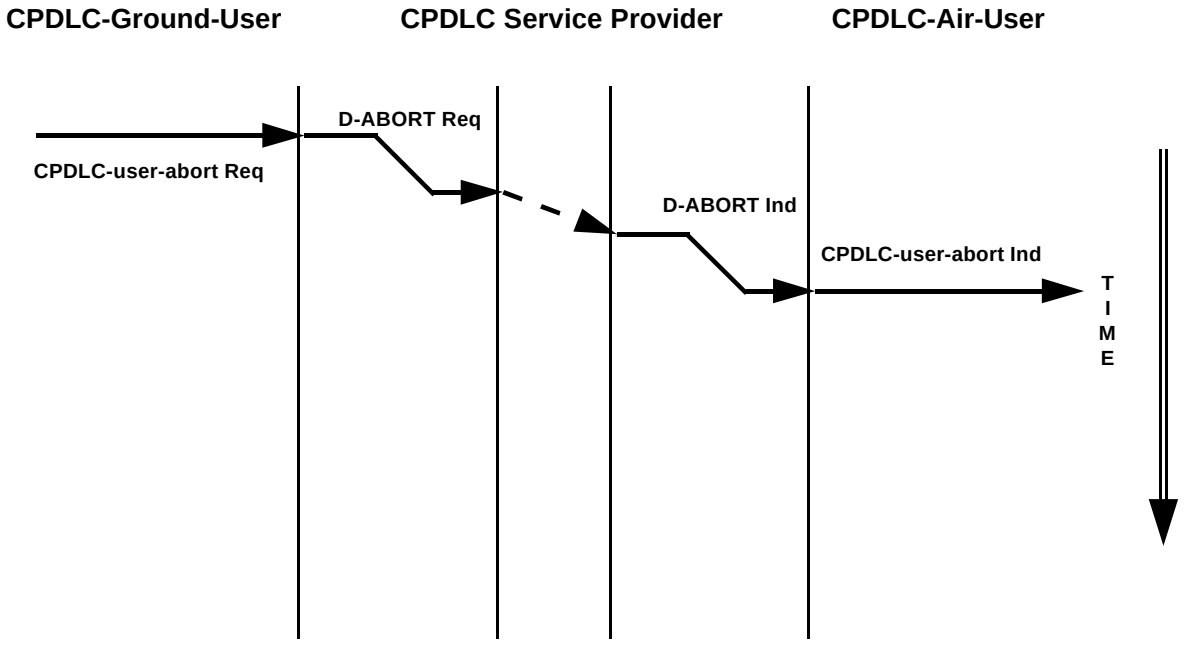

Figure 2.3.5-11. Sequence Diagram for CPDLC-user-abort Service/CPDLC-Ground-User Initiated

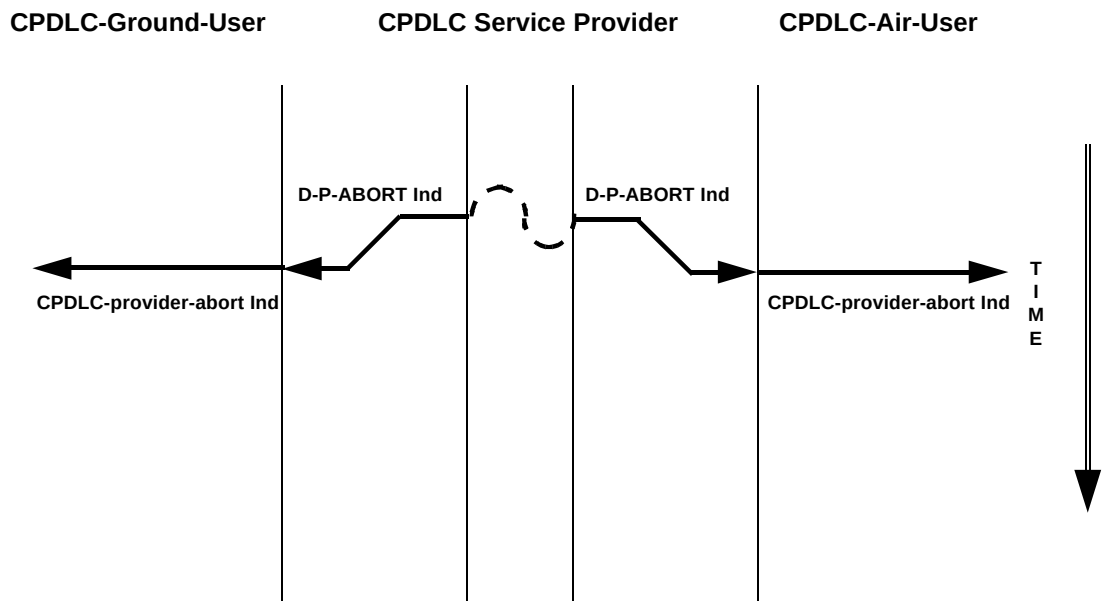

Figure 2.3.5-12. Sequence Diagram for CPDLC-provider-abort Service/Dialogue Service Abort

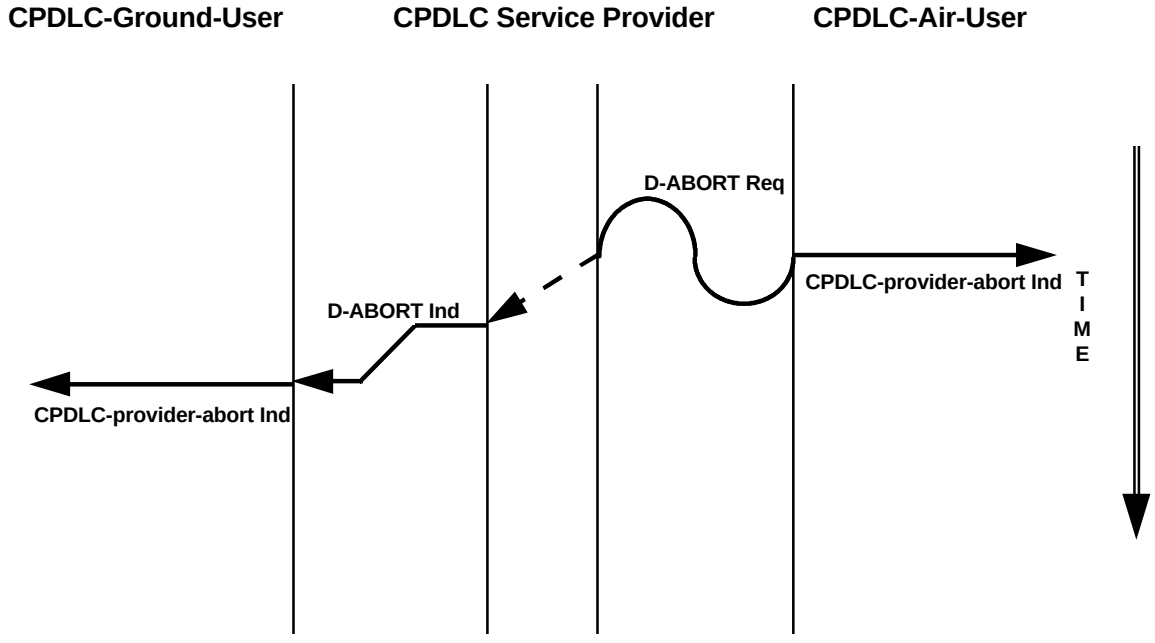

Figure 2.3.5-13. Sequence Diagram for CPDLC-provider-abort Service/CPDLC-Air-ASE Abort

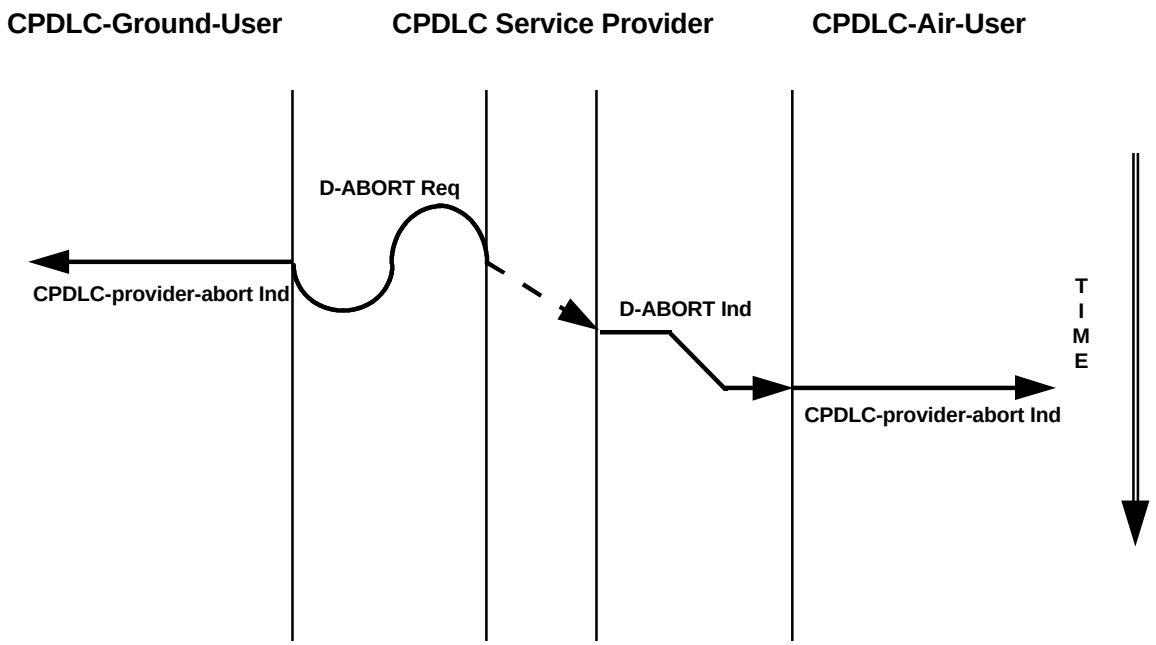

Figure 2.3.5-14. Sequence Diagram for CPDLC-provider-abort Service/ CPDLC-Ground-ASE Abort

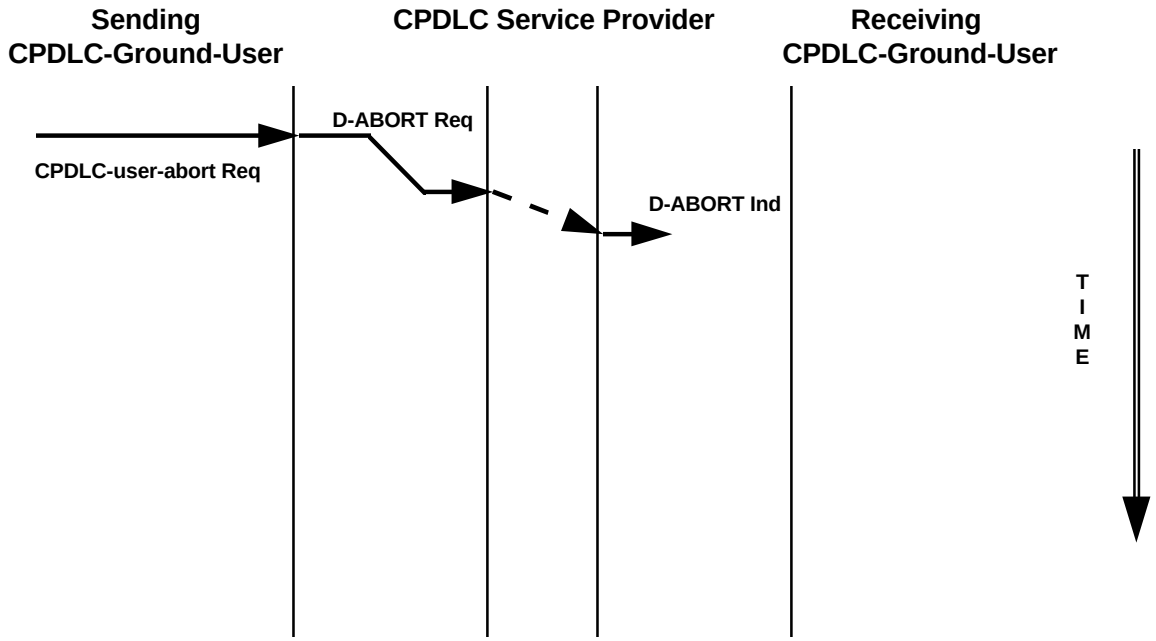

Figure 2.3.5-15. Sequence Diagram for CPDLC-user-abort Service/Sending CPDLC-Ground-User Initiated

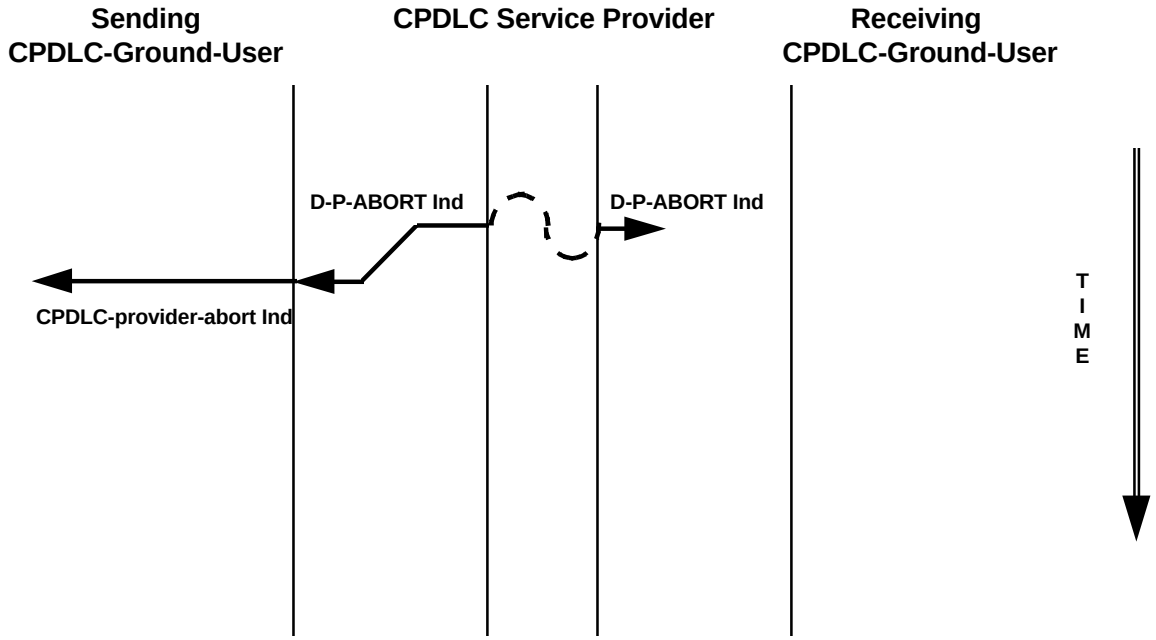

Figure 2.3.5-16. Sequence Diagram for CPDLC-provider-abort Service/Dialogue Service Abort

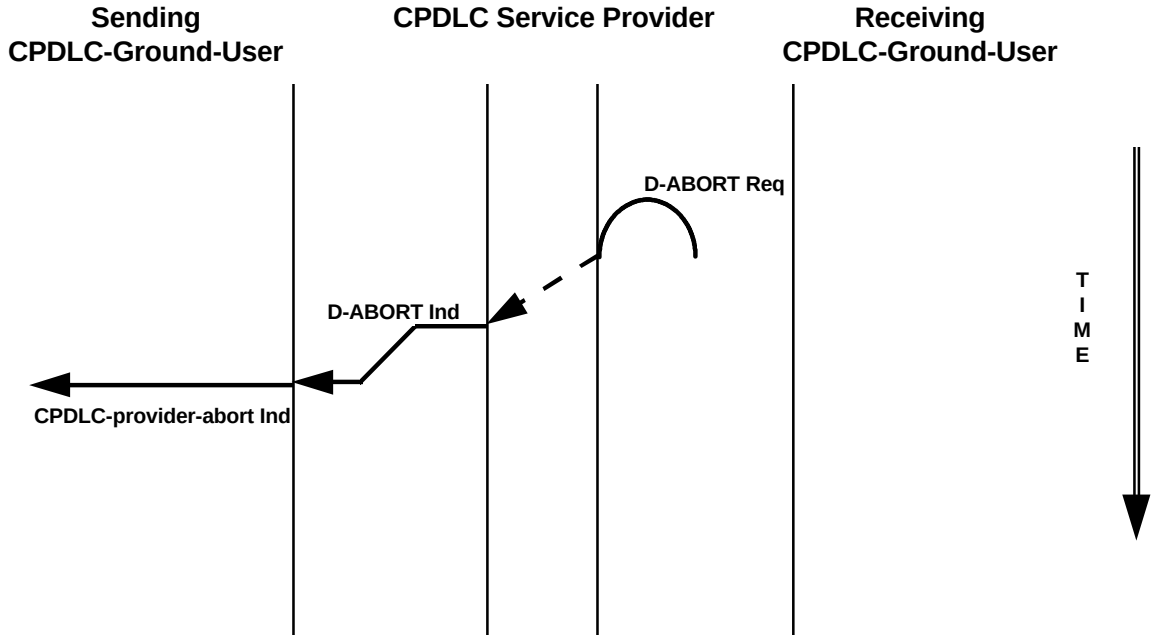

Figure 2.3.5-17. Sequence Diagram for CPDLC-provider-abort Service/Receiving CPDLC-Ground-ASE Abort

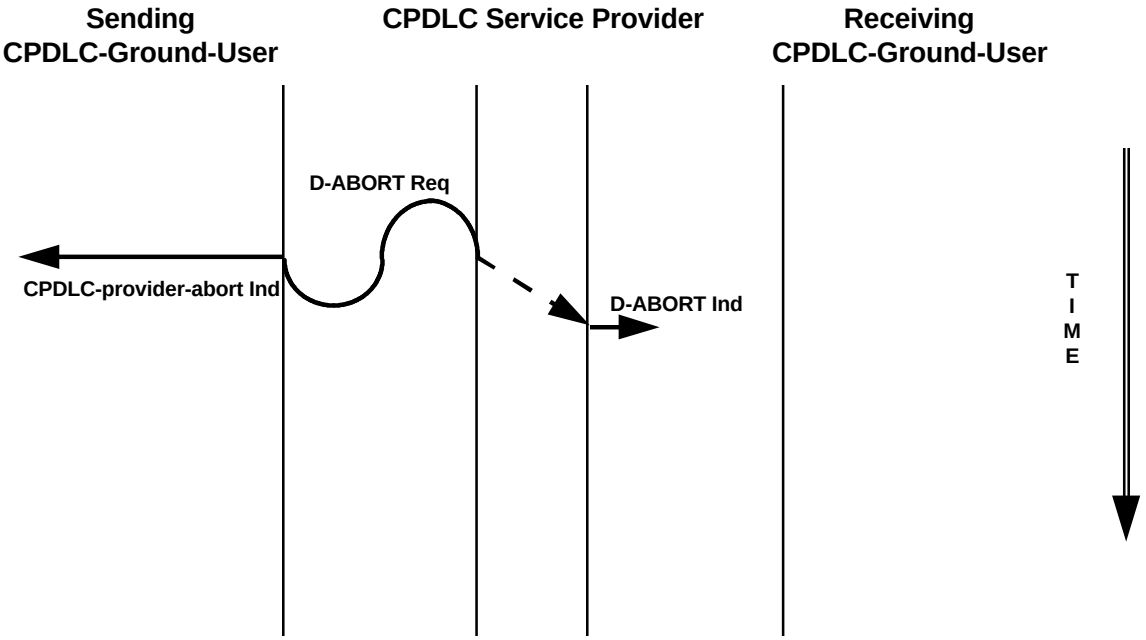

Figure 2.3.5-18. Sequence Diagram for CPDLC-provider-abort Service/Sending CPDLC-Ground-ASE Abort

2.3.5.2 CPDLC Service Provider Timers

2.3.5.2.1 A CPDLC-ASE shall be capable of detecting when a timer expires.

Note 1.— Table 2.3.5-1 lists the time constraints related to the CPDLC application. Each time constraint requires a timer to be set in the CPDLC protocol machine.

Note 2.— If the timer expires before the final event has occurred, a CPDLC-ASE takes appropriate action as defined in 2.3.5.4.1.

2.3.5.2.2 **Recommendation.** — *The timer values should be as indicated in Table 2.3.5-1.*

Table 2.3.5-1. CPDLC Service Provider Timers

CPDLC Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
CPDLC-start	t_{start}	6 minutes	D-START request	D-START confirmation
DSC-start	t_{start}	6 minutes	D-START request	D-START confirmation

CPDLC-forward	t_{start}	6 minutes	D-START request	D-START confirmation

Note.— The receipt of CPDLC-user-abort requests, D-ABORT Indications, or D-P-ABORT Indications are also timer stop events.

2.3.5.3 CPDLC-Air-ASE Protocol Description

2.3.5.3.1 Introduction

2.3.5.3.1.1 If no actions are described for a CPDLC service primitive when a CPDLC-air-ASE is in a specific state, then the invocation of that service primitive shall be prohibited while the CPDLC-air-ASE is in that state.

2.3.5.3.1.2 Upon receipt of a PDU, if no actions are described for the arrival of that PDU when a CPDLC-air-ASE is in a specific state, then that PDU is considered not permitted, and exception handling procedures as described in 2.3.5.4.4 shall apply.

2.3.5.3.1.3 If a PDU is received that cannot be decoded, then exception handling procedures as described in 2.3.5.4.3 for invalid PDU shall apply.

2.3.5.3.1.4 If a PDU is not received when one is required, then exception handling as described 2.3.5.4.3 shall apply.

Note 1.— The states defined for the CPDLC-air-ASE are the following.

a) IDLE

b) START-REQ,

c) START-IND,

d) DIALOGUE, and

e) END.

Note 2.— The CPDLC-air-user is an active user from:

a) the time it has invoked the CPDLC-start service request until:

1) receipt of a CPDLC-start service confirmation with Result parameter equal to the abstract value "rejected", or

2) invocation of a CPDLC-end service response with the Result parameter set to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

b) the time it has received the CPDLC-start service indication until:

1) invocation of a CPDLC-start service response with Result parameter equal to the abstract value "rejected", or

2) invocation of a CPDLC-end service response with the Result parameter set to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

c) the time it has invoked the DSC-start service request until:

1) receipt of a DSC-start service confirmation with Result parameter equal to the abstract value "rejected", or

2) receipt of a DSC-end service confirmation with Result parameter equal to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication.

2.3.5.3.1.5 On initiation the CPDLC-air-ASE shall be in the *IDLE* state.

Note.— The CPDLC-air-ASE contains a Boolean called DSC. DSC has the abstract value "true" when the dialogue is a DSC dialogue, and has the abstract value "false" otherwise.

2.3.5.3.1.6 On the initiation of a CPDLC-air-ASE, DSC shall be set to the abstract value "false".

2.3.5.3.2 D-START Indication

2.3.5.3.2.1 Upon receipt of a D-START indication, if the CPDLC-air-ASE is in the *IDLE* state and the D-START *User Data* parameter contains a GroundPDUs [UplinkMessage] APDU, and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety message" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low", and the D-START *QOS Routing Class* parameter has one of the abstract values specified in Table 2.3.6-1, the CPDLC-air-ASE shall:

a) Invoke CPDLC-start service indication containing the following:

1) the D-START *Calling Peer ID* parameter value as the CPDLC-start service *Calling Peer Identifier* parameter value,

2) the D-START *QOS Routing Class* parameter value as the CPDLC-start service *Class of Communication* parameter value,

3) if the GroundPDUs [UplinkMessage] APDU contained in the D-START *User Data* parameter is an ATCUplinkMessage, set the GroundPDUs APDU-element as the CPDLC-start service *CPDLC Message* parameter value, and

b) Enter the *START-IND* state.

2.3.5.3.3 D-START Confirmation

2.3.5.3.3.1 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "accepted" and DSC has the abstract value "false" and D-START *User Data* parameter is not provided, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation with the abstract value "accepted" as the CPDLC-start service *Result* parameter value,

c) Enter the *DIALOGUE* state.

2.3.5.3.3.2 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "rejected (permanent)" and the D-START *Reject Source* parameter has the abstract value "DS user" and DSC has the abstract value "false" and if the D-START *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:

1) if the D-START *User Data* parameter is provided, the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, and

2) the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.3.3.3 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true" and D-START *User Data* parameter is not provided, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke DSC-start service confirmation with the abstract value "accepted" as the DSC-start service *Result* parameter value,

c) Enter the *DIALOGUE* state.

2.3.5.3.3.4 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "rejected (permanent)" and the D-START *Reject Source* parameter has the abstract value "DS user", and DSC has the abstract value "true", and if the D-START *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke DSC-start service confirmation containing the following:

1) if the D-START *User Data* parameter is provided, the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, and

2) the abstract value "rejected" as the DSC-start service *Result* parameter value,

c) Set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.3.4 D-DATA Indication

2.3.5.3.4.1 Upon receipt of a D-DATA indication, if the CPDLC-air-ASE is in the *DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Invoke CPDLC-message service indication with the APDU contained in the D-DATA *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, and

b) Remain in the *DIALOGUE* state.

2.3.5.3.4.2 Upon receipt of a D-DATA indication, if the CPDLC-air-ASE is in the *END* state and DSC has the abstract value of "true" and the APDU contained in the D-DATA *User Data* parameter is a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Invoke CPDLC-message service indication with the APDU contained in the D-DATA *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, and

b) Remain in the *END* state.

2.3.5.3.5 D-END Indication

2.3.5.3.5.1 Upon receipt of a D-END indication, if the CPDLC-air-ASE is in the *DIALOGUE* state, and DSC has the abstract value "false", and if the D-END *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Invoke CPDLC-end service indication with the APDU contained in the D-END *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, if provided, as the CPDLC-end service *CPDLC Message* parameter value, and

b) Enter the *END* state.

2.3.5.3.6 D-END Confirmation

2.3.5.3.6.1 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the abstract the D-END *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true" and if the D-END *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Invoke DSC-end service confirmation with:

1) if the D-END *User Data* parameter is provided, the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, and

2) the abstract value “accepted” as the CPDLC-end service *Result* parameter value,

b) Set DSC to the abstract value “false”, and

c) Enter the *IDLE* state.

2.3.5.3.6.2 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the D-END *Result* parameter has the abstract value “rejected”, and DSC has the abstract value “true”, and if the D-END *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

a) Invoke DSC-end service confirmation with:

1) if the D-END *User Data* parameter is provided, the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, and

2) the abstract value “rejected” as the CPDLC-end service *Result* parameter value:

b) Enter the *DIALOGUE* state.

2.3.5.3.7 CPDLC-start Service Request

2.3.5.3.7.1 Upon receipt of a CPDLC-start service request, if the CPDLC-air-ASE is in the *IDLE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a StartDownMessage APDU element containing:

1) the abstract value “cpdlc” as the mode,

2) if provided, the *CPDLC Message* parameter as the DownlinkMessage, or

3) else, NULL as the DownlinkMessage,

b) Invoke D-START request with the following:

1) the CPDLC-start service *Called Peer Identifier* parameter value as the D-START *Called Peer ID* parameter value,

2) the CPDLC-start service *Calling Peer Identifier* parameter value as the D-START *Calling Peer ID* parameter value,

3) the D-START *Quality of Service* parameters set as follows:

i) if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START *QOS Routing Class* parameter value, or

ii) The abstract value of "high priority flight safety messages", as the D-START *QOS Priority* parameter value, and

iii) The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and

4) the APDU as the D-START *User Data* parameter value;

c) Start timer t_{start}, and

d) Enter the *START-REQ* state.

2.3.5.3.8 CPDLC-start Service Response

2.3.5.3.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-air-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "accepted" and the CPDLC-start service *Reject Reason* parameter is not provided, and DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) Invoke D-START response with the abstract value "accepted" as the D-START *Result* parameter value, and

b) Enter the *DIALOGUE* state.

2.3.5.3.8.2 Upon receipt of a CPDLC-start service response, if the CPDLC-air-ASE is in the *START-IND* state, and the CPDLC-start service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) If the CPDLC-start service *Reject Reason* parameter is provided, create an AircraftPDUs APDU with an ATCDownlinkMessage APDU element based on the *Reject Reason* parameter,

b) Invoke D-START response with the following:

1) if created, the APDU as the D-START *User Data* parameter, and

2) the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.3.9 DSC-start Service Request

2.3.5.3.9.1 Upon receipt of a DSC-start service request, if the CPDLC-air-ASE is in the *IDLE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a StartDownMessage APDU element containing:

1) the abstract value “dsc” as the mode,

2) if provided, the *CPDLC Message* parameter as the DownlinkMessage, or

3) else, NULL as the DownlinkMessage,

b) Invoke D-START request with the following:

1) the DSC-start service *Facility Designation* parameter value as the D-START *Called Peer ID* parameter value,

2) the DSC-start service *Aircraft Address* parameter value as the D-START *Calling Peer ID* parameter value,

3) Set the D-START *Quality of Service* parameters as follows:

i) if provided, the DSC-START service *Class of Communication* parameter value as the D-START *QOS Routing Class* parameter value,

ii) The abstract value of “high priority flight safety messages”, as the D-START *QOS Priority* parameter value, and

iii) The abstract value of “low” as the D-START *QOS Residual Error Rate* parameter value, and

4) the APDU as the D-START *User Data* parameter value;

c) Set DSC to the abstract value “true”,

d) Start timer t_{start}, and

e) Enter the *START-REQ* state.

2.3.5.3.10 CPDLC-message Service Request

2.3.5.3.10.1 Upon receipt of a CPDLC-message service request, if the CPDLC-air-ASE is in the *DIALOGUE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,

b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and

c) Remain in the *DIALOGUE* state.

2.3.5.3.10.2 Upon receipt of a CPDLC-message service request, if the CPDLC-air-ASE is in the *END* state and DSC has the abstract value “false”, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,

b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and

c) Remain in the *END* state.

2.3.5.3.11 CPDLC-end Service Response

2.3.5.3.11.1 Upon receipt of a CPDLC-end service response, if the CPDLC-air-ASE is in the END state, and the CPDLC-end service *Result* parameter has the abstract value “accepted” and DSC has the abstract value “false”, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,

b) Invoke D-END response with the following:

1) if created, the APDU as the D-END *User Data* parameter value, and

2) the abstract value “accepted”, as the D-END *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.3.11.2 Upon receipt of a CPDLC-end service response, if the CPDLC-air-ASE is in the *END* state, and the CPDLC-end service *Result* parameter has the abstract value “rejected” and DSC has the abstract value “false”, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,

b) Invoke D-END response with the following:

1) if created, the APDU as the D-END *User Data* parameter value, and

2) the abstract value “rejected”, as the D-END *Result* parameter value, and

c) Enter the *DIALOGUE* state.

2.3.5.3.12 DSC-end Service Request

2.3.5.3.12.1 Upon receipt of a DSC-end service request, if the CPDLC-air-ASE is in the *DIALOGUE* state and DSC has the abstract value “true”, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the DSC-end service *CPDLC Message* parameter, if provided,

b) Invoke D-END request with the APDU as the D-END *User Data* parameter value, if provided, and

c) Enter the *END* state.

2.3.5.3.13 CPDLC-user-abort Service Request

2.3.5.3.13.1 Upon receipt of a CPDLC-user-abort service request, if the CPDLC-air-ASE is not in the *IDLE* state, the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-user-abort service *Reason* parameter is provided, create an AircraftPDUs APDU with a CPDLCUserAbortReason APDU-element based on the CPDLC-user-abort service *Reason* parameter,

c) Else create an AircraftPDUs APDU with a CPDLCUserAbortReason [undefined] APDU-element,

d) Invoke D-ABORT request with the following:

1) the D-ABORT *Originator* parameter set to the abstract value “user”, and

2) the APDU as the D-ABORT *User Data* parameter value, and

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.3.14 D-ABORT Indication

2.3.5.3.14.1 Upon receipt of a D-ABORT indication, if the CPDLC-air-ASE is not in the *IDLE* state, and the D-ABORT *Originator* parameter is "user" and the D-ABORT *User Data* parameter contains a GroundPDUs [CPDLCUserAbortReason] APDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-user-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CPDLC-user-abort service *Reason* parameter value,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.3.14.2 Upon receipt of a D-ABORT indication, if the CPDLC-air-ASE is not in the *IDLE* state, and if the D-ABORT *Originator* parameter is "provider" and if the D-ABORT *User Data* parameter is provided, the D-ABORT *User Data* parameter contains a GroundPDUs [CPDLCProviderAbortReason] APDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.3.15 D-P-ABORT Indication

2.3.5.3.15.1 Upon receipt of a D-P-ABORT indication, if the CPDLC-air-ASE is not in the *IDLE* state, the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-failure",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4 CPDLC-Air-ASE Exception Handling

2.3.5.4.1 A Timer Expires

2.3.5.4.1.1 If a CPDLC-air-ASE detects that a timer has expired, that CPDLC-air-ASE shall:

a) Interrupt any current activity,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [timer-expired] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "timer-expired" as the CPDLC-provider abort service *Reason* parameter value,

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.4.2 Unrecoverable System Error

2.3.5.4.2.1 **Recommendation.** — *If a CPDLC-air-ASE has an unrecoverable system error, the CPDLC-air-ASE should:*

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [undefined-error] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT Originator parameter value, and

2) the APDU as the D-ABORT User Data parameter value, and

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "undefined-error" as the CPDLC-provider abort service Reason parameter value,

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the IDLE state.

2.3.5.4.3 Invalid PDU

2.3.5.4.3.1 If the *User Data* parameter of a D-END confirmation with *Result* parameter set to the abstract value "rejected", or if the *User Data* parameter of a D-START indication, a D-DATA indication, or a D-END indication, does not contain a valid PDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [invalid-PDU] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "invalid-PDU" as the CPDLC-provider abort service *Reason* parameter value,

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.4.3.2 If the *User Data* parameter of a D-START confirmation with *Result* set to the abstract value "rejected (permanent)", or a D-END confirmation with *Result* set to the abstract value "accepted", is not a valid PDU then the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "invalid-PDU" ,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4.4 Not Permitted PDU

2.3.5.4.4.1 If the *User Data* parameter of a D-START indication, D-DATA indication, or D-END indication is a valid PDU, but is not a PDU for which action is described within a given state in 2.3.5.3, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value “provider” as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU” as the CPDLC-provider abort service *Reason* parameter value ,

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.4.4.2 If a D-START confirmation with the *Result* parameter set to the abstract value “accepted” contains a *User Data* parameter the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value “provider” as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU” as the CPDLC-provider-abort service *Reason* parameter value,

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.4.4.3 If the *User Data* parameter of a D-END confirmation is a valid PDU, but is not a permitted PDU as defined in 2.3.5.3, the CPDLC-air-ASE shall:

a) If the D-END *Result* parameter is set to the abstract value "rejected", then

1) Stop any timer,

2) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,

3) Invoke D-ABORT request with:

i) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

ii) the APDU as the D-ABORT *User Data* parameter value, and

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider-abort service *Reason* parameter value,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4.5 D-START Confirmation *Result* or *Reject Source* Not as Expected

2.3.5.4.5.1 If a D-START confirmation *Result* parameter has the abstract value of "rejected (transient)" or if the *Reject Source* parameter has the abstract value of "DS provider", the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-error",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4.6 D-START Indication *Quality of Service* Not as Expected

2.3.5.4.6.1 If a D-START indication *QOS Priority* parameter does not have the abstract value of "high priority flight safety messages" or if the *QOS Residual Error Rate* parameter does not have the abstract value of "low", or if the *QOS Routing Class* parameter does not have one of the abstract values specified in Table 2.3.6-1, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as D-ABORT *User Data* parameter value,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4.7 Expected PDU Missing

2.3.5.4.7.1 If the *User Data* parameter of a D-START indication or D-DATA indication does not contain a PDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create on AircraftPDUs APDU with a CPDLCProviderAbortReason [expected-PDU-missing] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter values,

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider-abort service *Reason* parameter value,

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.5 CPDLC-Ground-ASE Protocol Description

2.3.5.5.1 Introduction

2.3.5.5.1.1 If no actions are described for a CPDLC service primitive when a CPDLC-ground-ASE is in specific state, then the invocation of that service primitive shall be prohibited while the CPDLC-ground-ASE is in that state.

2.3.5.5.1.2 Upon receipt of a PDU, if no actions are described for the arrival of that PDU when a CPDLC-ground-ASE is in a specific state, then that PDU is considered not permitted, and exception handling procedures as described in 2.3.5.6.4 shall apply.

2.3.5.5.1.3 If a PDU is received that cannot be decoded, then exception handling procedures as described in 2.3.5.6.3 for invalid PDU shall apply.

2.3.5.5.1.4 If a PDU is not received when one is required, then exception handling as described in 2.3.5.6.3 shall apply.

Note 1.— The states defined for the CPDLC-ground-ASE are the following.

a) IDLE

b) START-REQ,

c) START-IND,

d) DIALOGUE,

e) END, and

f) FORWARD.

Note 2.— The CPDLC-ground-user is an active user from:

a) the time it has invoked the CPDLC-start service request until:

1) receipt of a CPDLC-start service confirmation with Result parameter equal to the abstract value "rejected", or

2) receipt of a CPDLC-end service confirmation with the Result parameter equal to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of a CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

b) the time it has received the CPDLC-start service indication until:

1) invocation of a CPDLC-start service response with Result parameter set to the abstract value "rejected", or

2) receipt of a CPDLC-end service confirmation with the Result parameter equal to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

c) the time it has received the DSC-start service indication until:

1) invocation of a DSC-start service response with Result parameter equal to the abstract value "rejected", or

2) invocation of a CPDLC-user-abort service request, or

3) receipt of CPDLC-user-abort service indication, or

4) receipt of a CPDLC-provider-abort service indication; or

d) the time it has invoked the CPDLC-forward service request until:

1) receipt of a CPDLC-forward service confirmation,

2) invocation of a CPDLC-user-abort service request, or

3) receipt of CPDLC-user-abort service indication, or

4) receipt of a CPDLC-provider-abort service indication.

2.3.5.5.1.5 On initiation the CPDLC-ground-ASE shall be in the *IDLE* state.

Note.— The CPDLC-ground-ASE contains a Boolean called DSC. DSC has the abstract value "true" when the dialogue is a DSC dialogue, and has the abstract value "false" otherwise.

2.3.5.5.1.6 On the initiation of a CPDLC-ground-ASE, DSC shall be set to the abstract value "false".

2.3.5.5.2 D-START Indication

2.3.5.5.2.1 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a 24-bit aircraft address, and the D-START

User Data parameter contains an AircraftPDUs [StartDownMessage] APDU with the APDU-element mode "cpdlc", and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety messages" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low" and the D-START *QOS Routing Class* parameter has one of the abstract values specified in Table 2.3.6-1, the CPDLC-ground-ASE shall:

a) Invoke CPDLC-start service indication containing the following:

1) the D-START *Calling Peer ID* parameter value as the CPDLC-start service *Calling Peer Identifier* parameter value,

2) the D-START *QOS Routing Class* parameter value as the CPDLC-start service *Class of Communication* parameter value,

3) if the AircraftPDUs APDU-element contained in the D-START *User Data* parameter is an ATCDownlinkMessage, set the AircraftPDUs APDU-element as the CPDLC-start service *CPDLC Message* parameter value, and

b) Enter the *START-IND* state.

2.3.5.5.2.2 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a 24-bit aircraft address, and the D-START *User Data* parameter contains an AircraftPDUs [StartDownMessage]APDU with the APDU-element mode "dsc", and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety messages" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low", and the D-START *QOS Routing Class* parameter has one of the abstract values specified in Table 2.3.6-1 the CPDLC-ground-ASE shall:

a) Invoke DSC-start service indication containing the following:

1) the D-START *Calling Peer ID* parameter value as the DSC-start service *Aircraft Address* parameter value,

2) the D-START *QOS Routing Class* parameter value as the CPDLC-start service *Class of Communication* parameter value,

3) if the APDU AircraftPDUs APDU contained in the D-START is an ATCDownlinkMessage, set the AircraftPDUs APDU as the DSC-start service *CPDLC Message* parameter value,

b) Set DSC to "true", and

c) Enter the *START-IND* state.

2.3.5.5.2.3 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a Facility Designation, and the D-START *User Data* parameter contains a GroundPDUs [ATCForwardMessage] APDU and the CPDLC-ground-ASE supports the CPDLC-forward service, and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety messages" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low", the CPDLC-ground-ASE shall:

a) If the D-START *DS User Version Number* parameter value is equal to the CPDLC-ground-ASE version number:

1) Invoke CPDLC-forward service indication containing the following:

i) the D-START *Calling Peer ID* parameter value as the CPDLC-forward service *Calling Facility Designation* parameter value,

ii) set the D-START GroundPDUs APDU-element as the CPDLC-forward service *CPDLC Message* parameter value, and

2) Create a GroundPDUs APDU with an ATCForwardResponse [success] APDU element,

3) Invoke D-START response with the following:

i) the APDU as the D-START *User Data* parameter value, and

ii) the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and

4) Remain in the *IDLE* state.

b) If the D-START *DS User Version Number* parameter value is not equal to the CPDLC-ground-ASE version number:

1) Create a GroundPDUs APDU with an ATCForwardResponse [version-not-equal] APDU element,

2) Invoke D-START response with the following:

i) the CPDLC-ground-ASE version number as the D-START *DS User Version Number* parameter value,

ii) the APDU as the D-START *User Data* parameter value, and

iii) the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and

3) Remain in the *IDLE* state.

2.3.5.5.2.4 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a Facility Designation, and the D-START *User Data* parameter contains a GroundPDUs APDU and the APDU element is an ATCForwardMessage and the CPDLC-ground-ASE does not support the CPDLC-forward service, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCForwardResponse [service-not-supported] APDU element,

b) Invoke D-START response with the following:

1) the APDU as the D-START *User Data* parameter value, and

2) the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and

c) Remain in the *IDLE* state.

2.3.5.5.3 D-START Confirmation

2.3.5.5.3.1 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *START-REQ* state and if the D-START *Result* parameter has the abstract value "accepted", and DSC has the abstract value of "false" and D-START *User Data* parameter is not provided, the CPDLC-ground-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the abstract value "accepted" as the CPDLC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

2.3.5.5.3.2 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "rejected (permanent)" and the D-START *Reject Source* parameter has the abstract value "DS user" and DSC has the abstract value "false" and if the D-START *User Data* parameter is provided, the *User Data* parameter contains a AircraftPDUs [ATCDownlinkMessage] APDU, the CPDLC-ground-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:

1) if the D-START *User Data* parameter is provided, the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, and

2) the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.5.3.3 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *FORWARD* state and if the D-START *Result* parameter has the abstract value "rejected (permanent)" and the *Reject Source* parameter has the abstract value "DS user"and the D-START *User Data* parameter contains a GroundPDUs [ATCForwardResponse] APDU, the CPDLC-ground-ASE shall:

a) If the D-START *DS User Version Number* parameter value is equal to the CPDLC-ground-ASE version number:

1) Stop timer t_{start},

2) Invoke CPDLC-forward service confirmation with the D-START GroundPDUs APDU element as the CPDLC-forward service *Result* parameter value, and

3) Enter the *IDLE* state.

b) If the D-START *DS User Version Number* parameter value is not equal to the CPDLC-ground-ASE version number:

1) Stop timer t_{start},

2) Invoke CPDLC-forward service confirmation with the following:

i) the D-START GroundPDUs APDU-element as the CPDLC-forward service *Result* parameter value, and

ii) the D-START *DS User Version Number* parameter value as the CPDLC-forward service *ASE Version Number* parameter value, and

3) Enter the *IDLE* state.

2.3.5.5.4 D-DATA Indication

2.3.5.5.4.1 Upon receipt of a D-DATA indication, if the CPDLC-ground-ASE is in the *DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a AircraftPDUs [ATCDownlinkMessage] APDU, the CPDLC-ground-ASE shall:

a) Invoke CPDLC-message service indication with the APDU contained in the D-DATA *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, and

b) Remain in the *DIALOGUE* state.

2.3.5.5.4.2 Upon receipt of a D-DATA indication, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value "false" and the APDU contained in the D-DATA *User Data* parameter is an AircraftPDUs [ATCDownlinkMessage] APDU, the CPDLC-ground-ASE shall:

a) Invoke CPDLC-message service indication with the APDU contained in the D-DATA *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, and

b) Remain in the *END* state.

2.3.5.5.5 D-END Indication

2.3.5.5.5.1 Upon receipt of a D-END indication, if the CPDLC-ground-ASE is in the *DIALOGUE* state, and DSC has the abstract value "true", and if the D-END *User Data* parameter is provided, the *User Data* parameter contains an AircraftPDUs [ATCDownlinkMessage] APDU, the CPDLC-ground-ASE shall:

a) Invoke DSC-end service indication with the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, if provided, and

b) Enter the *END* state.

2.3.5.5.6 D-END Confirmation

2.3.5.5.6.1 Upon receipt of a D-END confirmation, if the CPDLC-ground-ASE is in the *END* state and the D-END *Result* parameter has the abstract value "accepted" and DSC has the abstract value "false" and if the D-END *User Data* parameter is provided, the *User Data* parameter contains an AircraftPDUs [ATCDownlinkMessage] APDU, the CPDLC-ground-ASE shall:

a) Invoke CPDLC-end service confirmation with:

1) The APDU contained in the D-END *User Data* parameter as the CPDLC-end service *CPDLC Message* parameter value, if provided, and

2) The abstract value "accepted" as the CPDLC-end service *Result* parameter value, and

b) Enter the *IDLE* state.

2.3.5.5.6.2 Upon receipt of a D-END confirmation, if the CPDLC-ground-ASE is in the *END* state and the D-END *Result* has the abstract value "rejected" and DSC has the abstract value "false", and if the D-END *User Data* parameter is provided, the *User Data* parameter contains an AircraftPDUs [ATCDownlinkMessage] APDU, the CPDLC-ground-ASE shall:

a) Invoke CPDLC-end service confirmation with:

1) The APDU contained in the D-END *User Data* parameter as the CPDLC-end service *CPDLC Message* parameter value, if provided, and

2) The abstract value "rejected" as the CPDLC-end service *Result* parameter value, and

b) Enter the *DIALOGUE* state.

2.3.5.5.7 CPDLC-start Service Request

2.3.5.5.7.1 Upon receipt of a CPDLC-start service request, if the CPDLC-ground-ASE is in the *IDLE* state, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an UplinkMessage APDU element containing:

1) if provided, the *CPDLC Message* parameter as the UplinkMessage, or

2) else, NULL as the UplinkMessage,

b) Invoke D-START request with the following:

1) the CPDLC-start service *Called Peer Identifier* parameter value as the D-START *Called Peer ID* parameter value,

2) the CPDLC-start service *Calling Peer Identifier* parameter value as the D-START *Calling Peer ID* parameter value,

3) the D-START *Quality of Service* parameters set as follows:

i) if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START *QOS Routing Class* parameter value,

ii) The abstract value of "high priority flight safety messages", as the D-START *QOS Priority* parameter value, and

iii) The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and

4) the APDU as the D-START *User Data* parameter value;

c) Start timer t_{start}, and

d) Enter the *START-REQ* state.

2.3.5.5.8 CPDLC-start Service Response

2.3.5.5.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "accepted" and the CPDLC-start service *Reject Reason* parameter is not provided, and DSC has the abstract value "false", the CPDLC-ground-ASE shall:

a) Invoke D-START response with the abstract value "accepted" as the D-START *Result* parameter value, and

b) Enter the *DIALOGUE* state.

2.3.5.5.8.2 Upon receipt of a CPDLC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "false", the CPDLC-ground-ASE shall:

a) If the CPDLC-start service *Reject Reason* parameter is provided, create a GroundPDUs APDU with an ATCUplinkMessage APDU element based on the *Reject Reason* parameter,

b) Invoke D-START response with the following:

1) The APDU as the D-START *User Data* parameter value; if the *Reject Reason* parameter was provided, and

2) the abstract value "rejected (permanent)"as the D-START *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.5.9 DSC-start Service Response

2.3.5.5.9.1 Upon receipt of a DSC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the DSC-start service *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true", the CPDLC-ground-ASE shall:

a) Invoke D-START response with the abstract value "accepted" as the D-START *Result* parameter value, and

b) Enter the *DIALOGUE* state.

2.3.5.5.9.2 Upon receipt of a DSC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the DSC-start service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "true", the CPDLC-ground-ASE shall:

a) If the DSC-start service *Reject Reason* parameter is provided, create a GroundPDUs APDU with an ATCUplinkMessage APDU element based on the *Reject Reason* parameter,

b) Invoke D-START response with the following:

1) The APDU element as D-START *User Data* parameter value; if the *Reject Reason* parameter was provided, and

2) the abstract value "rejected (permanent)"as the D-START *Result* parameter value,

c) Set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.5.10 CPDLC-message Service Request

2.3.5.5.10.1 Upon receipt of a CPDLC-message service request, if the CPDLC-ground-ASE is in the *DIALOGUE* state, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,

b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and

c) Remain in the *DIALOGUE* state.

2.3.5.5.10.2 Upon receipt of a CPDLC-message service request, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value "true", the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,

b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and

c) Remain in the *END* state.

2.3.5.5.11 CPDLC-end Service Request

2.3.5.5.11.1 Upon receipt of a CPDLC-end service request, if the CPDLC-ground-ASE is in the *DIALOGUE* state and DSC has the abstract value “false”, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,

b) Invoke D-END request with the APDU as the D-END *User Data* parameter value, if provided and,

c) Enter the *END* state.

2.3.5.5.12 DSC-end Service Response

2.3.5.5.12.1 Upon receipt of a DSC-end service response, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value “true”, and the DSC-end service *Result* parameter has the abstract value “accepted”, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the DSC-end service *CPDLC Message* parameter, if provided,

b) Invoke D-END response with the following:

1) the APDU as the D-END *User Data* parameter; if provided, and

2) the abstract value “accepted” as the D-END *Result* parameter value,

c) Set DSC to the abstract value “false”, and

d) Enter the *IDLE* state.

2.3.5.5.12.2 Upon receipt of a DSC-end service response, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value “true”, and the DSC-end service *Result* parameter has the abstract value “rejected”, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the DSC-end service *CPDLC Message* parameter, if provided,

b) Invoke D-END response with the following:

1) the APDU as the D-END *User Data* parameter; if provided, and

2) the abstract value “rejected” as the D-END *Result* parameter value, and

c) Enter the *DIALOGUE* state.

2.3.5.5.13 CPDLC-forward Service Request

2.3.5.5.13.1 Upon receipt of a CPDLC-forward service request, if the CPDLC-ground-ASE is in the *IDLE* state, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCForwardMessage APDU element based on the CPDLC-forward service *CPDLC Message* parameter,

b) Invoke D-START request with the following:

1) the CPDLC-forward service *Called Facility Designation* parameter value as the D-START *Called Peer ID* parameter value,

2) the CPDLC-start service *Calling Facility Designation* parameter value as the D-START *Calling Peer ID* parameter value,

3) the D-START *Quality of Service* parameters set as follows:

i) if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START *QOS Routing Class* parameter value,

ii) The abstract value of "high priority flight safety messages", as the D-START *QOS Priority* parameter value, and

iii) The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and

4) the APDU as the D-START *User Data* parameter value;

c) Start timer t_{start}, and

d) Enter the *FORWARD* state.

2.3.5.5.14 CPDLC-user-abort Service Request

2.3.5.5.14.1 Upon receipt of a CPDLC-user-abort service request, if the CPDLC-ground-ASE is not in the *IDLE* state, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-user-abort service *Reason* parameter is provided, create a GroundPDUs APDU with a CPDLCUserAbortReason APDU-element based on the CPDLC-user-abort service *Reason* parameter,

c) Else create a GroundPDUs APDU with a CPDLCUserAbortReason [undefined] APDU-element,

d) Invoke D-ABORT request with the following:

1) the D-ABORT *Originator* parameter set to the abstract value “user”, and

2) the APDU as the D-ABORT *User Data* parameter value, and

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.5.15 D-ABORT Indication

2.3.5.5.15.1 Upon receipt of a D-ABORT indication, if the CPDLC-ground-ASE is not in the *IDLE* state and the D-ABORT *Originator* parameter is “user” and the D-ABORT *User Data* parameter contains an AircraftPDUs [CPDLCUserAbortReason] APDU, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-ground-user is an active user, invoke CPDLC-user-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CPDLC-user-abort service *Reason* parameter value,

c) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

d) Enter the *IDLE* state.

2.3.5.5.15.2 Upon receipt of a D-ABORT indication, if the CPDLC-ground-ASE is not in the *IDLE* state, and if the D-ABORT *Originator* parameter is “provider” and if the D-ABORT *User Data* parameter is provided, the D-ABORT *User Data* parameter contains either an AircraftPDUs [CPDLCProviderAbortReason] APDU or a GroundPDUs [CPDLCProviderAbortReason] APDU , the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided,

c) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

d) Enter the *IDLE* state.

2.3.5.5.16 D-P-ABORT Indication

2.3.5.5.16.1 Upon receipt of a D-P-ABORT indication, if the CPDLC-ground-ASE is not in the *IDLE* state, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value “communication-service-failure”,

c) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

d) Enter the *IDLE* state.

2.3.5.6 CPDLC-Ground-ASE Exception Handling

2.3.5.6.1 A Timer Expires

2.3.5.6.1.1 If a CPDLC-ground-ASE detects that a timer has expired, that CPDLC-ground-ASE shall:

a) Interrupt any current activity,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [timer-expired] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value “provider” as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “timer-expired” as the CPDLC-provider abort service *Reason* parameter value”,

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.6.2 Unrecoverable System Error

2.3.5.6.2.1 **Recommendation.** — *If a CPDLC-ground-ASE has an unrecoverable system error, the CPDLC-ground-ASE should:*

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [undefined-error] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT Originator parameter value, and

2) the APDU as the D-ABORT User Data parameter value, and

d) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "undefined-error" as the CPDLC-provider abort service Reason parameter value",

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the IDLE state.

2.3.5.6.3 Invalid PDU

2.3.5.6.3.1 If the *User Data* parameter of a D-END confirmation with *Result* parameter set to the abstract value "rejected", a D-START indication, a D-DATA indication, or a D-END indication, does not contain a valid PDU, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [invalid-PDU] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "invalid-PDU" as the CPDLC-provider abort service *Reason* parameter value,

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.6.3.2 If the *User Data* parameter of a D-START confirmation with *Result* set to the abstract value "rejected (permanent)", or a D-END confirmation with *Result* set to the abstract value "accepted", is not a valid PDU the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "invalid-PDU",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.6.4 Not Permitted PDU

2.3.5.6.4.1 If the *User Data* parameter of a D-START indication, D-DATA indication, or a D-END indication is a valid PDU, but is not a PDU for which action is described within a given state as defined in 2.3.5.5, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider abort service *Reason* parameter value",

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.6.4.2 If D-START confirmation with the *Result* parameter set to the abstract value "accepted" contains a *User Data* parameter, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value “provider” as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU” as the CPDLC-provider-abort service *Reason* parameter value,

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.6.4.3 If the *User Data* parameter of a D-END confirmation is a valid PDU, but is not a permitted PDU for which action is described within a given state as defined in 2.3.5.5, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the D-END *Result* parameter is set to the abstract value “rejected”, then

1) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,

2) Invoke D-ABORT request with:

i) the abstract value “provider” as the D-ABORT *Originator* parameter value, and

ii) the APDU as the D-ABORT *User Data* parameter value, and

c) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU” as the CPDLC-provider-abort service *Reason* parameter value,

d) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

e) Enter the *IDLE* state.

2.3.5.6.5 D-START Confirmation Result or Reject Source Not as Expected

2.3.5.6.5.1 If a D-START confirmation *Result* parameter has the abstract value of "rejected (transient)" or if the *Reject Source* parameter has the abstract value of "DS provider", the CPDLC-ground-ASE shall:

a) Stop any timer,

b) if the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-error",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.6.6 D-START Indication Quality of Service Not as Expected

2.3.5.6.6.1 If a D-START indication *QOS Priority* parameter does not have the abstract value of "high priority flight safety messages" or if the *QOS Residual Error Rate* parameter does not have the abstract value of "low", or if the *QOS Routing Class* parameter does not have one of the abstract values specified in Table 2.3.6-1, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.6.7 Expected PDU Missing

2.3.5.6.7.1 If the *User Data* parameter of a D-START indication or a D-DATA indication does not contain a PDU, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [expected-PDU-missing] APDU message element,

c) Invoke D-ABORT request with:

1) the abstract value "provider" as the D-ABORT *Originator* parameter value, and

2) the APDU as the D-ABORT *User Data* parameter value, and

d) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider-abort service *Reason* parameter value,

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.7 CPDLC ASE State Tables

2.3.5.7.1 Priority

2.3.5.7.1.1 If the state tables shown for the CPDLC-air-ASE and the CPDLC-ground-ASE shown below conflict with textual statements made elsewhere in this document, the textual statements shall take precedence.

Note 1.— In the following state tables, the statement "cannot occur" means that if the implementation conforms to the SARPs, it is impossible for this event to occur. If the event does occur, this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE aborts with the error "unrecoverable system error".

Note 2.— In the following state tables, the statement "not permitted" means that the implementation must prevent this event from occurring through some local means. If the event does occur this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE performs a local rejection of the request rather than aborting the dialogue.

Table 2.3.5-2. CPDLC-Air-ASE State Table

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
Dialogue Service Events					
D-START Indication APDU = UplinkMessage	**!** CPDLC-start indication ⇒*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "false" No *User Data*	cannot occur	**!** Stop timer t_{start} **!** CPDLC-start confirmation ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and Reject Source "DS user", DSC="false" if *User Data*, APDU = ATCUplinkMessage	cannot occur	**!** Stop timer t_{start} **!** CPDLC-start confirmation ⇒*IDLE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "true" No *User Data*	cannot occur	**!** Stop timer t_{start} **!** DSC-start confirmation, ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and Reject Source "DS user", DSC= "true" if *User Data*, APDU = ATCUplinkMessage	cannot occur	**!** Stop timer t_{start} **!** DSC-start confirmation **!** Set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur
D-DATA Indication APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	**!** *CPDLC*-message indication ⇒*DIALOGUE*	**!** if DSC="true" **!** CPDLC-message indication ⇒*END* **!** else not permitted
D-END Indication: DSC="false" if *User Data*, APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	**!** CPDLC-end indication ⇒*END*	cannot occur
D-END Confirmation: DSC="true" *Result* "accepted" if *User Data*, APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	cannot occur	**!** DSC-end confirmation **!** Set DSC "false" ⇒*IDLE*

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
D-END Confirmation: DSC="true", *Result* "rejected" if *User Data*, APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	cannot occur	**!** DSC-end confirmation ⇒*DIALOGUE*
CPDLC-User Events					
CPDLC-start Request	**!** D-START request **!** Start timer t_{start} ⇒*START-REQ*	not permitted	not permitted	not permitted	not permitted
CPDLC-start Response DSC="false" *Result* "accepted"	not permitted	not permitted	**!** D-START response ⇒*DIALOGUE*	not permitted	not permitted
CPDLC-start Response DSC="false" *Result* "rejected"	not permitted	not permitted	**!** D-START response ⇒*IDLE*	not permitted	not permitted
DSC-start Request	**!** D-START request **!** set DSC ="true" **!** Start timer t_{start} ⇒*START-REQ*	not permitted	not permitted	not permitted	not permitted
CPDLC-message Request	not permitted	not permitted	not permitted	**!** D-DATA request ⇒*DIALOGUE*	**!** if DSC="false" **!** D-DATA request ⇒*END* **!** else not permitted
CPDLC-end Service Response DSC = "false" *Result* "accepted"	cannot occur	cannot occur	cannot occur	not permitted	**!** D-END response ⇒*IDLE*
CPDLC-end Service Response DSC = "false" *Result* "rejected"	cannot occur	cannot occur	cannot occur	not permitted	**!** D-END response ⇒*DIALOGUE*
DSC-end Request: DSC = "true"	not permitted	not permitted	not permitted	**!** D-END request ⇒*END*	not permitted
ABORT Events					
CPDLC-user-abort Request	not permitted	**!** Stop timer t_{start}, if set **!** D-ABORT request **!** If DSC = "true", set DSC = "false" ⇒*IDLE*	**!** Stop timer t_{start}, if set **!** D-ABORT request **!** If DSC = "true", set DSC = "false" ⇒*IDLE*	**!** D-ABORT request **!** If DSC = "true", set DSC = "false" ⇒*IDLE*	**!** D-ABORT request **!** If DSC = "true", set DSC = "false" ⇒*IDLE*

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
D-ABORT Indication *Originator* "user"	cannot occur	! Stop timer T_{start}, if set ! If active user: CPDLC-user-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	! Stop timer T_{start}, if set ! If active user: CPDLC-user-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	! If active user: CPDLC-user-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	! If active user: CPDLC-user-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*
D-ABORT Indication *Originator* "provider"	cannot occur	! Stop timer T_{start}, if set ! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC "false" ⇒*IDLE*	! Stop timer T_{start}, if set ! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC "false" ⇒*IDLE*	! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC "false" ⇒*IDLE*	! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC "false" ⇒*IDLE*
D-P-ABORT indication	cannot occur	! Stop timer t_{start}, if set ! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	! Stop timer t_{start}, if set ! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	! If active user: CPDLC-provider-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*
T_{start} Expires	cannot occur	! D-ABORT request ! CPDLC-provider-abort indication ! If DSC = "true", set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur

Table 2.3.5-3. CPDLC-Ground-ASE State Table

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
Dialogue Service Events						
D-START Indication APDU Aircraft mode "cpdlc"	**!** CPDLC-start indication ⇒*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication APDU Aircraft mode "dsc"	**!** DSC-start indication, **!** Set DSC ="true" ⇒*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication APDU Forward version equal and function supported	**!** CPDLC-forward indication **!** D-START response ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication APDU Forward version not equal or function not supported	**!** D-START response ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted" DSC = "false"	cannot occur	**!** Stop timer t_{start} **!** CPDLC-start confirmation ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user" DSC="false"	cannot occur	**!** Stop timer t_{start} **!** CPDLC-start confirmation ⇒*IDLE*	cannot occur	cannot occur	cannot occur	**!** Stop timer t_{start} **!** CPDLC-forward confirmation ⇒*IDLE*
D-START Confirmation *Result* "accepted" DSC ="true"	cannot occur	**!** Stop timer t_{start} **!** DSC-start confirmation, ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user" DSC = "true"	cannot occur	**!** Stop timer t_{start} **!** DSC-start confirmation **!** Set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
D-DATA Indication	cannot occur	cannot occur	cannot occur	**!** CPDLC-message indication ⇒*DIALOGUE*	**!** if DSC = "false **!** CPDLC-message indication ⇒*END* **!** else not permitted	cannot occur
D-END Indication: DSC="true"	cannot occur	cannot occur	cannot occur	**!** DSC-end indication ⇒*END*	cannot occur	cannot occur
D-END Confirmation: DSC = "false" *Result* "accepted"	cannot occur	cannot occur	cannot occur	cannot occur	**!** CPDLC-end confirmation ⇒*IDLE*	cannot occur
D-END Confirmation: DSC = "false" *Result* "rejected"	cannot occur	cannot occur	cannot occur	cannot occur	**!** CPDLC-end confirmation ⇒*DIALOGUE*	cannot occur
CPDLC-User Events						
CPDLC-start Request	**!** D-START request **!** Start timer t_{start} ⇒*START-REQ*	not permitted	not permitted	not permitted	not permitted	not permitted
CPDLC-start Response DSC = "false" *Result* "accepted"	not permitted	not permitted	**!** D-START response ⇒*DIALOGUE*	not permitted	not permitted	not permitted
CPDLC-start Response DSC = "false" *Result* "rejected"	not permitted	not permitted	**!** D-START response ⇒*IDLE*	not permitted	not permitted	not permitted
DSC-start Response DSC = "true" *Result* "accepted"	not permitted	not permitted	**!** D-START response ⇒*DIALOGUE*	not permitted	not permitted	not permitted
DSC-start Response DSC = "true" *Result* "rejected"	not permitted	not permitted	**!** D-START response **!** Set DSC= "false" ⇒*IDLE*	not permitted	not permitted	not permitted

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
CPDLC-message Request	not permitted	not permitted	not permitted	**!** D-DATA request ⇒*DIALOGUE*	**!** If DSC= "true" **!** D-DATA request ⇒*END* **!** Else not permitted	not permitted
CPDLC-end Request: DSC = "false"	not permitted	not permitted	not permitted	**!** D-END request ⇒*END*	not permitted	not permitted
DSC-end Service Response DSC = "true" *Result* "accepted"	cannot occur	cannot occur	cannot occur	not permitted	**!** D-END response **!** Set DSC = "false" ⇒*IDLE*	not permitted
DSC-end Service Response DSC = "true" *Result* "rejected"	cannot occur	cannot occur	cannot occur	not permitted	**!** D-END response ⇒*DIALOGUE*	not permitted
CPDLC-forward Request	**!** D-START request **!** Start timer t_{start} ⇒*FORWARD*	not permitted	not permitted	not permitted	not permitted	not permitted
ABORT Events						
CPDLC-user-abort Request	not permitted	**!** Stop timer t_{start}, if set **!** D-ABORT request **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer t_{start}, if set **!** D-ABORT request **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** D-ABORT request **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** D-ABORT request **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer t_{start}, if set **!** D-ABORT request ⇒*IDLE*
D-ABORT Indication *Originator* "provider"	cannot occur	**!** Stop timer T_{start}, if set **!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer T_{start}, if set **!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer T_{start}, if set **!** If active user: CPDLC-provider-abort indication ⇒*IDLE*

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
D-ABORT Indication *Originator* "user"	cannot occur	**!** Stop timer T_{start}, if set **!** If active user: CPDLC-user-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer T_{start}, if set **!** If active user: CPDLC-user-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** If active user: CPDLC-user-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** If active user: CPDLC-user-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer T_{start}, if set **!** If active user: CPDLC-user-abort indication ⇒*IDLE*
D-P-ABORT indication	cannot occur	**!** Stop timer t_{start}, if set **!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer t_{start}, if set **!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** If active user: CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	**!** Stop timer t_{start}, if set **!** If active user: CPDLC-provider-abort indication ⇒*IDLE*
T_{start} Expires	cannot occur	**!** D-ABORT request **!** CPDLC-provider-abort indication **!** If DSC = "true", set DSC= "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur	**!** D-ABORT request **!** CPDLC-provider -abort indication ⇒*IDLE*

2.3.6 COMMUNICATION REQUIREMENTS

2.3.6.1 Encoding Rules

2.3.6.1.1 The CPDLC application shall use PER as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in 2.3.4.

2.3.6.2 Dialogue Service Requirements

2.3.6.2.1 Primitive Requirements

2.3.6.2.1.1 Where dialogue service primitives, that is D-START, D-DATA, D-END, D-ABORT, and D-P-ABORT are described as being invoked in 2.3.5, the CPDLC-ground-ASE and the CPDLC-air-ASE shall exhibit external behavior consistent with the dialogue service, as described in 4.2 having been implemented and its primitives invoked.

2.3.6.2.2 Quality-of-Service Requirements

2.3.6.2.2.1 The application service priority for CPDLC shall have the abstract value of “high priority flight safety messages”.

2.3.6.2.2.2 The RER Quality of Service Parameter of the D-START shall be set to the abstract value of “low”.

2.3.6.2.2.3 The CPDLC-ASE shall map the CPDLC-start service or DSC-start service *Class of Communication* parameter abstract value to the ATSC routing class abstract value part of the D-START QOS parameter as presented in Table 2.3.6-1.

Table 2.3.6-1. Mapping Between Class of Communication and Routing Class Abstract Values

Class of Communication Abstract Value	Routing Class Abstract Value
A	Traffic follows Class A ATSC route(s)
B	Traffic follows Class B ATSC route(s)
C	Traffic follows Class C ATSC route(s)
D	Traffic follows Class D ATSC route(s)
E	Traffic follows Class E ATSC route(s)
F	Traffic follows Class F ATSC route(s)
G	Traffic follows Class G ATSC route(s)
H	Traffic follows Class H ATSC route(s)

Note.— ATSC values are defined in 1.3.

2.3.7 CPDLC USER REQUIREMENTS

2.3.7.1 General

Note 1.— Requirements imposed on CPDLC-user concerning CPDLC messages and interfacing with the CPDLC-ASEs are presented in this chapter.

Note 2.— Where reference is made to the "CPDLC-user", this implies both the CPDLC-air-user and the CPDLC-ground-user.

Note 3.— A CPDLC message is:

a) what the CPDLC-user provides the CPDLC-service as the CPDLC Message or Reject Reason parameter, when invoking a CPDLC service request or response primitive, or

b) what the CPDLC-user receives in the same parameters from the CPDLC service indication or confirmation primitives.

Note 4.— The terms CPDLC message, message, uplink message and downlink message are used interchangeably, and equate to a CPDLC message. When the terms "send" and "transmit" are used this means that the CPDLC-user has invoked a CPDLC service request or response primitive. When the term "receive" is used this means that a CPDLC indication or confirmation primitive parameter containing a CPDLC message has been provided by the CPDLC service.

Note 5.— If there are any differences and/or inconsistencies between material contained in this chapter and those contained in the in the Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services *(PANS-RAC, Doc 4444), the latter would take precedence.*

2.3.7.1.1 General CPDLC Service Requirements

2.3.7.1.1.1 A CPDLC-ground-user shall invoke CPDLC-start service, DSC-start service, CPDLC-message service, CPDLC-end service, and DSC-end service only when communicating with a CPDLC-air-user.

2.3.7.1.1.2 A CPDLC-ground-user shall invoke CPDLC-forward service, only when communicating with another CPDLC-ground-user.

Note 1.— When a CPDLC-user invokes the CPDLC-start service, the DSC-start service, or the CPDLC-forward service and requires a particular class of communication service, the CPDLC-user sets the Class of Communication Service parameter.

Note 2.— When a CPDLC-user does not require a particular class of communication, the user does not set the Class of Communication Service parameter.

2.3.7.2 CPDLC Message Generation Requirements

Note 1.— A response message is a message which is a reply to a received message. It contains a message reference number identical to the message identification number of the message to which it refers. Only response messages contain a message reference number.

Note 2.— Message response attributes dictate a) if a response is required or prohibited; b) if a response is required, dictate the permitted response messages.

Note 3.— A closure response message is a reply to a message or series of messages which terminates a sequence of message exchanges. However due to the multiple element capability of a CPDLC message, a closure message may contain message element(s) in addition to the required closure message element that initiate a new sequence of messages.

2.3.7.2.1 Message Composition

2.3.7.2.1.1 A CPDLC message shall be composed of a message header, and from one to five message elements.

2.3.7.2.1.2 For air/ground messages, the message header shall be composed of a message identification number, a message reference number, if required, a time stamp, and a logical acknowledgment requirement (optional).

2.3.7.2.1.3 For ground/ground messages, the message header shall be composed of a time stamp, the aircraft flight identification, and the aircraft address to which the message refers.

2.3.7.2.1.4 A message element shall consist of a message element identifier, data as indicated by the specified message element, and associated message element attributes.

2.3.7.2.1.5 For each CPDLC message the CPDLC-user sends air/ground it shall provide the following information:

a) a message identification number,

b) a message reference number only if the message is a response message,

c) date and time,

d) a logical acknowledgment indication, if required,

e) from one to five message element identifiers, and

f) data as required for each message element identification included.

2.3.7.2.1.6 For each CPDLC message the CPDLC-user sends ground/ground it shall provide the following information:

a) the aircraft identification to which the ground/ground message refers,

b) date and time,

c) from one to five message element identifiers, and

d) data as required for each message element identification included.

2.3.7.2.2 Message Identification Number

Note 1.— A message identification number pertains to a single peer to peer dialogue.

Note 2.— Message identification numbers used by a CPDLC ground system for uplink messages to an aircraft have no relationship to the message identification numbers used by the same ground system another aircraft.

Note 3.— Similarly, message identification numbers used by a CPDLC aircraft for downlink messages to a CPDLC ground system have no relationship to the message identification numbers used by the same aircraft with another ground system.

Note 4.— There is no relationship between message identification numbers assigned and managed by a the CPDLC ground system and those message identification numbers assigned and managed by the aircraft.

2.3.7.2.2.1 The message identification number provided by the CPDLC-user shall be different from any other message identification number currently in use.

2.3.7.2.2.2 A message identification number shall be deemed currently in use until:

a) if the message does not allow a response, the message is sent, or

b) if the message requires a response, the closure response is received.

2.3.7.2.2.3 If a CPDLC or DSC dialogue is terminated, all message identification numbers pertaining to that dialogue shall be considered available.

2.3.7.2.3 Message Elements That Cannot Be Combined With Other Message Elements in a Message

2.3.7.2.3.1 The LOGICAL ACKNOWLEDGMENT message element (uplink message element 227 and downlink message element 100) shall be sent only as a single message element CPDLC message.

2.3.7.2.3.2 The NEXT DATA AUTHORITY [facility] message element (uplink message element 160) shall be sent only as a single message element CPDLC message.

2.3.7.2.4 Restriction on Route Clearance Variable Message Elements

2.3.7.2.4.1 A CPDLC message shall contain no more than two message elements with the [routeClearance] variable.

2.3.7.3 CPDLC Message Receipt Requirements

2.3.7.3.1 Message Attributes

Note 1.— Message attributes dictate certain message handling requirements for the CPDLC-user receiving a message. Each CPDLC message has Urgency, Alert, and Response attributes.

Note 2.— Message element attribute table entries are listed in order of precedence (i.e., a precedence value of 1 is highest followed by 2, etc.).

2.3.7.3.1.1 When a message contains a single message element, the message attributes shall be the message element attributes.

2.3.7.3.1.2 When a message contains multiple message elements, the highest precedence message element attribute for each attribute type associated with any element in the message shall be the message attribute for each attribute type for the entire message.

2.3.7.3.2 Urgency Requirements

Note.— The Urgency (URG) attribute delineates the queuing requirements for received messages that are displayed to the end-user. The same Urgency attribute types are used for both air/ground and ground/ground messages.

2.3.7.3.2.1 Each message element shall have associated Urgency attributes with precedence as defined in table 2.3.7-1.

Table 2.3.7-1. Urgency Attribute (Air/Ground and Ground/Ground)

Type	Description	Precedence
D	Distress	1
U	Urgent	2
N	Normal	3
L	Low	4

2.3.7.3.2.2 When a CPDLC-user queues received messages, messages with the highest Urgency type shall be placed at the beginning of the queue.

2.3.7.3.2.3 When a CPDLC-user queues received messages, messages with the same Urgency type shall be queued in order of receipt.

2.3.7.3.3 Alerting Requirements

Note.— The alert (ALRT) attribute delineates the type of end-user alerting required by the CPDLC-user upon message receipt. The same Alert attribute types are used for both air/ground and ground/ground messages.

2.3.7.3.3.1 Each message element shall have associated Alert attributes with precedence as defined in table 2.3.7-2.

Table 2.3.7-2. Alert Attribute (Air/Ground and Ground/Ground)

Type	Description	Precedence
H	High	1
M	Medium	2
L	Low	3
N	No alerting required	4

2.3.7.3.3.2 Upon receipt of a CPDLC message, the CPDLC-user shall provide one of three distinct alerts as determined by the received message alert attribute.

2.3.7.3.4 Response Attribute

Note.— The response (RESP) attribute mandates CPDLC-user response requirements for a given message element. Response message attribute only apply to air/ground messages.

2.3.7.3.4.1 Each uplink message element shall have associated Response attributes with precedence as defined in table 2.3.7-3.

Table 2.3.7-3. Response Attribute (Up-Link)

Type	***Response Required***	***Valid Responses Description***	***Precedence***
W/U	*Yes*	*Response required: WILCO, UNABLE, STANDBY permitted, LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary)*	*1*
A/N	*Yes*	*Response required: AFFIRM, NEGATIVE, STANDBY permitted,LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary)*	*2*

R	*Yes*	*Response required: ROGER, UNABLE, STANDBY permitted, LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary)*	*3*
Y	*Yes*	*Any CPDLC downlink message, LOGICAL ACKNOWLEDGMENT (only if required),*	*4*
N	*No, unless logical acknowledgment required*	*LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary, only when logical acknowledgment is required)*	*5*

2.3.7.3.4.2 Each downlink message element shall have associated Response attributes with precedence as defined in table 2.3.7-4.

Table 2.3.7-4. Response Attribute (Down-Link)

Type	***Response Required***	***Valid Responses Description***	***Precedence***
Y	*Yes*	*Any CPDLC uplink message, LOGICAL ACKNOWLEDGMENT (only if required)*	*1*
N	*No, unless logical acknowledgment required*	*LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary, only when logical acknowledgment is required)*	*2*

2.3.7.3.5 CPDLC/DSC Distinction

2.3.7.3.5.1 Upon receipt of a CPDLC message the CPDLC-user shall provide a distinction between CPDLC messages received from the Current Data Authority and those received from a Downstream Data Authority.

2.3.7.3.6 Air/Ground - Ground/Ground Distinction

2.3.7.3.6.1 Upon receipt of a CPDLC message the CPDLC-user shall provide a distinction between CPDLC messages received from an aircraft and those received from another ground system.

2.3.7.3.7 Logical Acknowledgment Prohibited

2.3.7.3.7.1 Upon receipt of the CPDLC message USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED the CPDLC-air-user shall be prohibited from requiring a logical acknowledgment for any message sent for the duration of the CPDLC or DSC dialogue.

2.3.7.3.7.2 If the CPDLC-ground-user receives a CPDLC message requiring a logical acknowledgment where the use of logical acknowledgment has been prohibited as above, the CPDLC-ground-user shall invoke the CPDLC-message service with a message containing the ERROR [errorinformation] message element with the [logicalAcknowledgmentNotAccepted] value as the CPDLC Message parameter and discard the content of the received message.

2.3.7.3.8 Message Reference Numbers

2.3.7.3.8.1 If a received message requires a response, the CPDLC-user shall provide a message reference number for each response message sent.

2.3.7.3.8.2 The message reference number shall be identical to the message identification number of the received message to which it refers.

2.3.7.3.9 Message Response Requirements

2.3.7.3.9.1 **Recommendation.** — *A message sequence initiated by data link should be closed by data link.*

2.3.7.3.9.2 **Recommendation.** — *If a message sequence exchange initiated by data link is subsequently closed by voice, local procedures should be in place to ensure deletion of outstanding data link messages requiring closure.*

2.3.7.3.9.3 A CPDLC-user shall only be permitted to respond to a received message in its entirety.

2.3.7.3.9.4 Only one closure response shall be permitted for a given message.

2.3.7.3.9.5 If the CPDLC-air-user has not issued a DSC-end service request primitive, or if the CPDLC-air-user has issued a DSC-end service request primitive for which a DSC-end service confirmation primitive has been received with the *Result* parameter containing the abstract value “rejected” then, if a message is received that requires a response, the CPDLC-air-user shall either:

a) send any permitted response messages and then send a closure response message, or

b) send a closure response message.

2.3.7.3.9.6 If the CPDLC-ground-user has not issued a CPDLC-end service request primitive, or if the CPDLC-ground-user has issued a CPDLC-end service request primitive for which a CPDLC-end service confirmation primitive has been received with the *Result* parameter containing the abstract value “rejected” then, if a message is received that requires a response, the CPDLC-ground-user shall either:

a) send any permitted response messages and then send a closure response message, or

b) send a closure response message.

2.3.7.3.9.7 For a given message, once the CPDLC-user has sent the closure response message, no other response messages shall be sent referring to the given message.

2.3.7.3.9.8 When a message is received by the CPDLC-user requiring a logical acknowledgment response, the CPDLC-user shall respond with either a CPDLC message containing a LOGICAL ACKNOWLEDGMENT message element, or a message containing an ERROR message element.

2.3.7.3.9.9 A logical acknowledgment response message, if required, shall be sent prior to sending any other related response message(s), except a response message containing an ERROR message element.

2.3.7.3.9.10 When the CPDLC-air-user receives a message with a W/U RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, WILCO, UNABLE, or ERROR message element.

2.3.7.3.9.11 When the CPDLC-air-user receives a message with a W/U RESP attribute, the closure response message shall contain at least a WILCO, UNABLE, or ERROR message element.

2.3.7.3.9.12 When the CPDLC-air-user receives a message with an A/N RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, AFFIRM, NEGATIVE, or ERROR message element.

2.3.7.3.9.13 When the CPDLC-air-user receives a message with an A/N RESP attribute, the closure response message shall contain at least a AFFIRM, NEGATIVE, or ERROR message element.

2.3.7.3.9.14 When the CPDLC-air-user receives a message with a R RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, ROGER, UNABLE or ERROR message element.

2.3.7.3.9.15 When the CPDLC-air-user receives a message with a R RESP attribute, the closure response message shall contain at least a ROGER, UNABLE or ERROR message element.

2.3.7.3.9.16 When the CPDLC-air-user receives a message with a Y RESP attribute, a LOGICAL ACKNOWLEDGMENT only when requested, and all other CPDLC messages shall be permitted as a response message.

2.3.7.3.9.17 When the CPDLC-air-user receives a message with a Y RESP attribute, the first response message sent that does not contain a STANDBY or LOGICAL ACKNOWLEDGMENT shall constitute the closure response message.

2.3.7.3.9.18 When the CPDLC-ground-user receives an air/ground message with a Y RESP attribute, a LOGICAL ACKNOWLEDGMENT, only when requested and all other CPDLC messages shall be permitted as a response message.

2.3.7.3.9.19 When the CPDLC-ground-user receives an air/ground message with a Y RESP attribute, the first response message sent that does not contain a STANDBY, REQUEST DEFERRED, or LOGICAL ACKNOWLEDGMENT message element shall constitute the closure response message.

2.3.7.3.9.20 When the CPDLC-air-user receives a message with a N RESP attribute, but requiring a logical acknowledgment, the only permitted response shall be a message that contains a LOGICAL ACKNOWLEDGMENT or ERROR message element. This response message is the closure message.

2.3.7.3.9.21 When the CPDLC-ground-user receives an air/ground message with a N RESP attribute, but requiring a logical acknowledgment the only permitted response shall be a message that contains a LOGICAL ACKNOWLEDGMENT or ERROR message element. This response message is the closure message.

2.3.7.3.9.22 When the CPDLC-ground-user receives a ground/ground message the ground-user shall be prohibited from generating a ground/ground response message.

Note.— Ground/ground forwarding of messages is a one-way exchange on a one message per dialogue basis. There are no message identification or message reference numbers contained in the header of a ground/ground message.

2.3.7.3.10 Invalid Message Elements

2.3.7.3.10.1 The CPDLC-ground-user shall be prohibited from sending any CPDLC message containing the uplink message elements 33, 40, 41, or 178.

2.3.7.3.11 Error Conditions

2.3.7.3.11.1 Duplicate Message Identification Numbers

2.3.7.3.11.1.1 If a CPDLC message is received containing an identification number identical to that of an identification number currently in use the CPDLC-user shall invoke the CPDLC-user-abort request service with a CPDLC message containing the CPDLCUserAbortReason with value [duplicate-message-identification-number] as the *Reason* parameter.

2.3.7.3.11.2 Invalid Reference Number

2.3.7.3.11.2.1 If the CPDLC-user receives a message containing a message reference number which is not identical to any message identification number currently in use, the CPDLC-user shall:

a) invoke CPDLC-message request with the ERROR [errorinformation] message element with the [unrecognizedMsgReferenceNumber] value as the CPDLC Message parameter, and

b) disregard the received message.

2.3.7.3.11.3 No Available Message Identification Numbers

2.3.7.3.11.3.1 If the CPDLC-user attempts to send a CPDLC message and all message identification numbers are currently in use, the CPDLC-user shall invoke the CPDLC-user-abort request with a CPDLC

message containing the CPDLCUserAbortReason with the value [no-message-identification-numbers-available] as the *Reason* parameter.

2.3.7.3.11.4 Insufficient Resources

2.3.7.3.11.4.1 If the CPDLC-user receives a message and has insufficient resources to handle the message, the CPDLC-user shall:

a) invoke CPDLC-message request with the ERROR [errorinformation] message element with the [insufficientResources] value as the *CPDLC Message* parameter, and

b) disregard the received message.

2.3.7.3.11.5 Invalid Message Element Combination

2.3.7.3.11.5.1 If a message is received containing:

a) a LOGICAL ACKNOWLEDGMENT message element in combination with any other message element in a single message,

b) a NEXT DATA AUTHORITY [facility] message element in combination with any other message element in a single message, or

c) a message containing more than two message elements with the [routeClearance] variable,

the CPDLC-user shall invoke the CPDLC-message request with the ERROR [errorinformation] message element eith the [invalidMessageElementCombination] value as the *CPDLC Message* parameter, and disregard the received message.

2.3.7.3.11.6 Invalid Message Elements

2.3.7.3.11.6.1 If the CPDLC-air-user receives a message containing any of the uplink message element identifiers 33, 40, 41 or 178 the CPDLC-air-user shall invoke the CPDLC-message request with the ERROR [errorinformation] message element with the [invalidMessageElement] value as the *CPDLC Message* parameter, and disregard the received message.

2.3.7.3.11.7 ERROR Message Response

2.3.7.3.11.7.1 If the CPDLC-user sends a message containing the ERROR message element instead of the expected response message, the ERROR message shall contain the received message identification number as the message reference number.

2.3.7.3.11.7.2 If an ERROR message is sent in response to a CPDLC message the received CPDLC message shall be disregarded.

2.3.7.3.11.8 Invalid Message Response

2.3.7.3.11.8.1 If the CPDLC-ground-user sends a message that has a W/U response attribute, and a response to this message is received by the CPDLC-ground-user that does not contain any of the following message elements: WILCO, UNABLE, STANDBY, LOGICAL ACKNOWLEDGMENT, or ERROR[errorInformation] the CPDLC-ground-user shall invoke CPDLC-user-abort request with a CPDLC message containing the CPDLCUserAbortReason with the value [invalid-response].

2.3.7.3.11.8.2 If the CPDLC-ground-user sends a message that has an A/N response attribute, and a response to this message is received by the CPDLC-ground-user that does not contain any of the following message elements: AFFIRM, NEGATIVE, STANDBY, LOGICAL ACKNOWLEDGMENT, or ERROR[errorInformation] the CPDLC-ground-user shall invoke CPDLC-user-abort request with a CPDLC message containing the CPDLCUserAbortReason with the value [invalid-response].

2.3.7.3.11.8.3 If the CPDLC-ground-user sends a message that has a R response attribute, and a response to this message is received by the CPDLC-ground-user that does not contain any of the following message elements: ROGER, UNABLE, STANDBY, LOGICAL, ACKNOWLEDGMENT, or ERROR [errorInformation] the CPDLC-ground-user shall invoke CPDLC-user-abort request with a CPDLC message containing the CPDLCUserAbortReason with the value [invalid-response].

2.3.7.3.11.8.4 If the CPDLC-user sends a message that has a N response attribute and requires a logical acknowledgment, and a response to this message is received by the CPDLC-user that does not contain any of the following message elements: LOGICAL ACKNOWLEDGMENT, or ERROR[errorInformation] the CPDLC-user shall invoke CPDLC-user-abort request with a CPDLC message containing the CPDLCUserAbortReason with the value [invalild-response].

2.3.7.4 CPDLC-air-user Requirements

2.3.7.4.1 The CPDLC-start Service

2.3.7.4.1.1 Invoking the CPDLC-start request

2.3.7.4.1.1.1 If there is no CPDLC service, the only CPDLC service primitives the CPDLC-air-user shall be permitted to invoke are the CPDLC-start request or the DSC-start request.

2.3.7.4.1.1.2 The CPDLC-air-user shall only be permitted to invoke the CPDLC-start request with:

a) any ground system, if there is no existing CPDLC service for the CPDLC-air-user, or

b) the Next Data Authority, if the CPDLC-air-user has received a message from the Current Data Authority designating a Next Data Authority.

2.3.7.4.1.1.3 If a CPDLC-air-user has invoked a CPDLC-start request, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive on the CPDLC dialogue initiated by the CPDLC-start service request, except the CPDLC-user-abort request, until after it has received a CPDLC-start confirmation.

2.3.7.4.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response

2.3.7.4.1.2.1 Upon receipt of a CPDLC-start service indication, the CPDLC-user shall invoke a CPDLC-start service response within 0.5 seconds.

2.3.7.4.1.2.2 Upon receipt of a CPDLC-start indication the CPDLC-air-user shall invoke the CPDLC-start response, with the response parameters set as follows:

a) The *Result* parameter to the abstract value of "accepted" if:

   1) There is no existing CPDLC service, or

   2) CPDLC service exists and the request is from either the Current Data Authority or Next Data Authority,

b) Else set the *Result* parameter is set to the abstract value "rejected" and the *Reject Reason* to a CPDLC message with the message element NOT AUTHORIZED NEXT DATA AUTHORITY.

2.3.7.4.1.2.3 If a CPDLC-start indication is received from either the Current Data Authority or the Next Data Authority, and this results in a second CPDLC dialogue being established with a given ground system, the CPDLC-air-user shall invoke the CPDLC-user-abort request primitive for the first connection with that ground system.

2.3.7.4.1.2.4 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value "rejected" any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be disregarded.

2.3.7.4.1.2.5 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value "accepted" and the request is from the Current Data Authority any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be processed.

2.3.7.4.1.2.6 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value "accepted" and the request is from the Next Data Authority any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be disregarded.

2.3.7.4.1.2.7 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value "accepted" the CPDLC-air-user shall:

a) Establish an association between a CPDLC-ASE invocation and a ground system facility designation contained in CPDLC-start indication *Calling Peer Identifier* parameter,

b) If there is no Current Data Authority, associate this CPDLC-ASE invocation with the Current Data Authority, or

c) If the facility designation contained CPDLC-start indication *Calling Peer Identifier* parameter is the Next Data Authority associate the CPDLC-ASE invocation with the Next Data Authority.

2.3.7.4.1.2.8 If CPDLC-start indication has been received, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive with this ground system, except the CPDLC-user-abort request, until after it has invoked the CPDLC-start response.

2.3.7.4.1.3 Receipt of a CPDLC-start confirmation

2.3.7.4.1.3.1 If a CPDLC-start confirmation has been received with a *Result* parameter containing the abstract value "accepted" the CPDLC-air-user shall:

a) Establish an association between a CPDLC-ASE invocation and a ground system facility designation contained in CPDLC-start request *Called Peer Identifier* parameter,

b) If there is no Current Data Authority, associate the CPDLC-ASE invocation with the Current Data Authority, or

c) If the facility designation contained CPDLC-start indication *Called Peer Identifier* parameter is the Next Data Authority associate the CPDLC-ASE invocation with the Next Data Authority.

2.3.7.4.2 The DSC-start Service

2.3.7.4.2.1 Invoking the DSC-start request

2.3.7.4.2.1.1 Only a CPDLC-air-user shall be permitted to invoke the DSC-start service request primitive.

2.3.7.4.2.1.2 A CPDLC-air-user shall only be permitted to invoke the DSC-start-service request primitive if the CPDLC-air-user has no existing DSC dialogue.

2.3.7.4.2.1.3 If a CPDLC-air-user has invoked a DSC-start request, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive on the DSC dialogue initiated by the DSC-start service request except the CPDLC-user-abort request, until after it has received a DSC-start confirmation.

2.3.7.4.2.2 Receipt of a DSC-start confirmation

2.3.7.4.2.2.1 If a DSC-start confirmation has been received with a *Result* parameter containing the abstract value "accepted" the CPDLC-air-user shall:

a) Establish an association between a CPDLC-ASE invocation and the ground system facility designation contained in DSC-start request *Facility Designation* parameter,

b) Associate the CPDLC-ASE invocation with a Downstream Data Authority.

2.3.7.4.3 The CPDLC-message Service

2.3.7.4.3.1 Receipt of a CPDLC-message Indication

2.3.7.4.3.1.1 Upon receipt of a CPDLC-message indication, if the indication is from the Current Data Authority or a Downstream Data Authority the CPDLC-air-user shall process the CPDLC message contained in the *CPDLC Message* parameter.

2.3.7.4.3.1.2 If a CPDLC-message indication is received from the Current Data Authority containing only the uplink message element NEXT DATA AUTHORITY specifying a facility as the Next Data Authority the CPDLC-air-user shall do the following in the order listed:

a) Check that there is no other Next Data Authority already established;

b) if there is, invoke CPDLC-user-abort request with the established Next Data Authority with the *Reason* parameter set to CPDLCUserAbortReason value [no-longer-next-data-authority]; and

c) then designate the ground system indicated in the CPDLC message from the CPDLC-message indication as the Next Data Authority.

2.3.7.4.3.1.3 If a CPDLC-message indication is received from the Current Data Authority containing only the uplink message element NEXT DATA AUTHORITY, indicating a “NULL” for the Next Data Authority the CPDLC-air-user shall do the following in the order listed:

a) check if there is a Next Data Authority already established;

b) if there is, invoke CPDLC-user-abort request with the established Next Data Authority with the *Reason* parameter set to CPDLCUserAbortReason value [no-longer-next-data-authority]; and

c) cancel any existing Next Data Authority designation.

2.3.7.4.3.1.4 If a CPDLC-message indication is received containing the uplink message element NEXT DATA AUTHORITY, and it is not from the Current Data Authority the message shall be disregarded.

2.3.7.4.3.1.5 Upon receipt of a CPDLC-message indication, if the indication is not from the Current Data Authority or a Downstream Data Authority, the CPDLC-air-user shall:

a) Invoke CPDLC-message service request with the *CPDLC Message* parameter containing a message with the message element NOT CURRENT DATA AUTHORITY, and

b) Disregard the received message contained in the CPDLC-message indication *CPDLC Message* parameter.

2.3.7.4.4 The CPDLC-end Service

2.3.7.4.4.1 The CPDLC-end Service Request

2.3.7.4.4.1.1 The CPDLC-air-user shall be prohibited from invoking the CPDLC-end request.

2.3.7.4.4.2 Receipt of a CPDLC-end Indication and Invoking a CPDLC-end Response

2.3.7.4.4.2.1 If a CPDLC-end indication is received but it is not from the Current Data Authority the CPDLC-air-user shall:

a) Invoke CPDLC-end response with

1) the *CPDLC Message* parameter containing a message with the message element NOT CURRENT DATA AUTHORITY, and

2) the *Result* parameter set to the abstract value “rejected”, and

b) Disregard any message provided in the CPDLC-end indication *CPDLC Message* parameter.

Note 1.— A CPDLC-air-user is considered to have uplink open messages when the CPDLC-air-user has received a message(s) for which a response is required, and it has not yet sent the closure response to the message(s).

Note 2.— Uplink open messages are not considered in setting out the requirements for the CPDLC-end Service. The ground user is aware of any such messages, and desires to end the dialogue anyway. Any such messages are considered deleted upon transmission of a CPDLC-end response with the Result parameter set to the abstract value “accepted” on the airborne side, and upon receipt of a CPDLC-end-confirmation with the Result parameter set to the abstract value “accepted” on the ground side.

Note 3.— A CPDLC-air-user is considered to have downlink open messages when the CPDLC-air-user has sent any messages that require a response for which it has not received a closure response.

Note 4.— Downlink open message are considered deleted upon transmission of a CPDLC-end response with the Result parameter set to the abstract value “accepted” on the airborne side, and upon receipt of a CPDLC-end confirmation with the Result paramter set to the abstract value “accepted” on the ground side.

Note 5.— Local procedures may dictate when a "rejected" response Result paramter is permitted by the CPDLC-air-user in the cases when these SARPs give the CPDLC-air-user a choice between "accepted" or "rejected".

Note 6.— If a CPDLC-end-service response is invoked with a "accepted" Result this will result in ending the dialogue regardless of any messages contained in the CPDLC Message parameter.

2.3.7.4.4.2.2 Uplink Message Contains Error

2.3.7.4.4.2.2.1 If a CPDLC-end indication is received from the Current Data Authority and there is a message in the *CPDLC Message* parameter that either has the response attribute N and requires a logical acknowledgment or has a W/U, A/N, R, or Y response attribute, and an error is detected in the message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *CPDLC Message* paramter containing a CPDLC message with the message element ERROR [errorInformation],

b) the *Result* parameter set to the abstract value "rejected".

2.3.7.4.4.2.3 No Message in the CPDLC-end Indication

2.3.7.4.4.2.3.1 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user does not have any downlink open messages and there is no message in the *CPDLC Message* parameter, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value "accepted".

2.3.7.4.4.2.3.2 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC -air-user has downlink open messages and there is no message in the *CPDLC Message* parameter, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.4.4.2.4 Message in CPDLC-end Indication Does Not Require Response

2.3.7.4.4.2.4.1 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user does not have any downlink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value "accepted".

2.3.7.4.4.2.4.2 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has downlink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.4.4.2.5 Message in CPDLC-end Indication Requires Only Logical Acknowledgment

2.3.7.4.4.2.5.1 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user does not have any downlink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *CPDLC Message* parameter containing a CPDLC message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value “accepted”.

2.3.7.4.4.2.5.2 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has downlink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *CPDLC Message* parameter containing a CPDLC message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value “accepted” or “rejected”.

2.3.7.4.4.2.6 Message in CPDLC-end Indication With W/U, A/N, or R Attribute, Positive Response

Note.— In this case, positive response to a message in the CPDLC-end Indication also indicates acceptance of the end of the dialogue.

2.3.7.4.4.2.6.1 If a CPDLC-end indication is received from the Current Data Authority and there is a message in the *CPDLC Message* parameter with the response attribute W/U, A/N or R and no error is detected in the message, then if the CPDLC-air-user chooses to respond positively (WILCO, AFFIRM, or ROGER) to the message, then the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *CPDLC Message* parameter containing a CPDLC message with at least the message element WILCO, AFFIRM, or ROGER as appropriate, and

2) the *Result* parameter set to the abstract value “accepted”.

2.3.7.4.4.2.7 Message in CPDLC-end Indication With W/U, A/N, or R Attribute, Negative Response

Note.— In this case, negative response to a message in the CPDLC-end Indication also indicates rejection of the end of the dialogue.

2.3.7.4.4.2.7.1 If a CPDLC-end indication is received from the Current Data Authority and there is a message in the *CPDLC Message* parameter with the response attribute W/U, A/N or R and no error is detected in the message, then if the CPDLC-air-user chooses to respond negatively (UNABLE or NEGATIVE) to the message, then the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *CPDLC Message* parameter containing a CPDLC message with at least the message element UNABLE or NEGATIVE as appropriate, and

2) the *Result* parameter set to the abstract value “rejected”.

2.3.7.4.4.2.8 Message in CPDLC-end Indication With Y Response Attribute

2.3.7.4.4.2.8.1 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user does not have any downlink open messages and there is a message in the *CPDLC Message* parameter with the response attribute Y and no error is detected in the message, the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *CPDLC Message* parameter containing a Y attribute closure CPDLC message, and

2) the *Result* parameter set to the abstract value “accepted”.

2.3.7.4.4.2.8.2 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has any downlink open messages and there is a message in the *CPDLC Message* parameter with the response attribute Y and no error is detected in the message, the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *CPDLC Message* parameter containing a Y attribute closure CPDLC message, and

2) the *Result* parameter set to the abstract value “accepted”.

2.3.7.4.4.2.9 Upon invoking CPDLC-end response with Result parameter set to “accepted”, the CPDLC-air-user shall:

a) delete any association with a ground system and Current Data Authority, and

b) if a ground system is designated as Next Data Authority and an association with a CPDLC-ASE exists, replace the Next Data Authority association with a Current Data Authority association, or

c) if a ground system is designated as Next Data Authority and no association with a CPDLC-ASE exists, delete Next Data Authority association with any ground system.

2.3.7.4.4.2.10 If the CPDLC-air-ASE associated with the Current Data Authority ceases to exist for any reason other than in response to a CPDLC-end request as specified above, any existing Next Data Authority designation and/or association shall cease to exist.

2.3.7.4.5 The DSC-end Service

2.3.7.4.5.1 The DSC-end Request

2.3.7.4.5.1.1 Only the CPDLC-air-user shall be permitted to invoke the DSC-end request.

2.3.7.4.5.1.2 If a CPDLC-air-user has invoked a DSC-end service request primitive, the air-user shall be prohibited from invoking any CPDLC service primitive with this ground system (except the CPDLC-user-abort request primitive) until it receives a DSC-end service confirmation primitive.

2.3.7.4.6 The CPDLC-user-abort Service

2.3.7.4.6.1 Issuing a CPDLC-user-abort Request [commanded-termination]

2.3.7.4.6.1.1 The CPDLC-air-user shall have the capability to invoke CPDLC-user-abort request with the *Reason* parameter set to CPDLCUserAbortReason value [commanded-termination].

2.3.7.4.6.2 Invoking a CPDLC-user-abort Request

2.3.7.4.6.2.1 If the CPDLC-air-user invokes CPDLC-user-abort request with the Current Data Authority, the CPDLC-air-user shall:

a) Delete any association of a ground system to a Current Data Authority,

b) If a ground system is designated as Next Data Authority and an association with a CPDLC-ASE exists, invoke CPDLC-user-abort request with the *Reason* parameter set to the value [current-data authority-abort] and

c) Delete any association of a ground system to a Next Data Authority.

2.3.7.4.6.2.2 If the CPDLC-air-user invokes CPDLC-user-abort request with the Next Data Authority, and then does not set the *Reason* parameter as [current-data-authority-abort] or [no-longer-next-data-authority], the CPDLC-air-user shall continue to maintain the association of the ground system to the Next Data Authority.

2.3.7.4.6.3 Receipt of a CPDLC-abort Indication

2.3.7.4.6.3.1 If the CPDLC-air-user receives a CPDLC-user-abort indication from the Current Data Authority or a CPDLC-provider-abort indication that causes the ASE invocation associated with the Current Data Authority to cease to exist, the CPDLC-air-user shall:

a) Delete any association of a ground system to a Current Data Authority,

b) If a ground system is designated as Next Data Authority and an association with a CPDLC-ASE exists, invoke CPDLC-user-abort request with the *Reason* parameter set to the value [current-data authority-abort], and

c) Delete any association of a ground system to a Next Data Authority.

2.3.7.4.6.3.2 If the CPDLC-air-user receives a CPDLC-user-abort indication from the Next Data Authority or receives a CPDLC-provider-abort indication that causes the ASE invocation associated with the Next Data Authority to cease to exist, the CPDLC-air-user shall continue to maintain the association of the ground system to the Next Data Authority.

2.3.7.5 CPDLC-Ground-User Requirements

2.3.7.5.1 The CPDLC-start Service

2.3.7.5.1.1 Invoking the CPDLC-start request

2.3.7.5.1.1.1 If there is no CPDLC service, the only CPDLC service primitives the CPDLC-ground-user shall be permitted to invoke are the CPDLC-start-request or the CPDLC-forward request.

2.3.7.5.1.1.2 If a CPDLC-ground-user has invoked a CPDLC-start request, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive on the CPDLC dialogue intiated by the CPDLC-start service request, except the CPDLC-user-abort request until after it has received a CPDLC-start confirmation.

2.3.7.5.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response

2.3.7.5.1.2.1 If a CPDLC-start indication is received from an aircraft with which the ground system currently has a CPDLC dialogue, the CPDLC-ground-user shall:

a) invoke the CPDLC-start response with the *Result* parameter set to the abstract value “accepted”, and

b) invoke the CPDLC-user-abort request for the first CPDLC dialogue with that aircraft.

2.3.7.5.1.2.2 The CPDLC-ground-user shall be prohibited from invoking the CPDLC-start response unless and until it has received a CPDLC-start indication.

2.3.7.5.1.2.3 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value “rejected” then the *Reject Reason* shall be an uplink message containing only the SERVICE UNAVAILABLE message element.

2.3.7.5.1.2.4 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value “rejected” the CPDLC-ground-user shall disregard any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter.

2.3.7.5.1.2.5 If the CPDLC-ground-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be processed.

2.3.7.5.1.2.6 If the CPDLC-ground-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in CPDLC-start indication *Calling Peer Identifier* parameter.

2.3.7.5.1.2.7 If CPDLC-start indication has been received, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive, except the CPDLC-user-abort request with that aircraft, until after it has invoked the CPDLC-start response.

2.3.7.5.1.3 Receipt of a CPDLC-start confirmation

2.3.7.5.1.3.1 If a CPDLC-start confirmation has been received with a *Result* parameter containing the abstract value "accepted" the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in CPDLC-start request *Called Peer Identifier* parameter.

2.3.7.5.2 The DSC-start Service

2.3.7.5.2.1 Receipt of a DSC-start Indication and Invoking DSC-start Response

2.3.7.5.2.1.1 The CPDLC-ground-user shall be prohibited from invoking the DSC-start response unless and until it has received a DSC-start indication.

2.3.7.5.2.1.2 If a DSC-start indication is received from an aircraft with which the ground system currently has a DSC dialogue, the CPDLC-ground-user shall:

a) invoke the DSC-start response with the *Result* parameter set to the abstract value "accepted", and

b) invoke the CPDLC-user-abort request for the first DSC dialogue with that aircraft.

2.3.7.5.2.1.3 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "rejected" then the *Reject Reason* shall be an uplink message containing only the SERVICE UNAVAILABLE message element.

2.3.7.5.2.1.4 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "rejected" the CPDLC-ground-user shall disregard any CPDLC message contained in the DSC-start indication *CPDLC Message* parameter.

2.3.7.5.2.1.5 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "accepted" any CPDLC message contained in the DSC-start indication *CPDLC Message* parameter shall be processed.

2.3.7.5.2.1.6 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "accepted" the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in the DSC-start indication *Aircraft Address* parameter.

2.3.7.5.2.1.7 If DSC-start indication has been received, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive, except the CPDLC-user-abort request, until after it has invoked the DSC-start response.

2.3.7.5.2.2 Receipt of a DSC-start Indication and Invoking a DSC-start Response

2.3.7.5.2.2.1 Upon receipt of a DSC-start indication, the CPDLC-ground-user shall invoke a DSC-start response within 0.5 seconds.

2.3.7.5.2.2.2 Upon receipt of a DSC-start indication, the CPDLC-ground-user shall invoke a DSC-start response, with the response parameters set as follows:

a) the *Result* parameter set to the abstract value “accepted” or “rejected”, and

b) if, and only if, the *Result* parameter is set to the abstract value “rejected”, then set the *Reject Reason* parameter to a CPDLC message with the message element SERVICE UNAVAILABLE.

2.3.7.5.3 The CPDLC-end Service

2.3.7.5.3.1 The CPDLC-end Request

2.3.7.5.3.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-end request.

2.3.7.5.3.1.2 If a CPDLC-ground-user has invoked a CPDLC-end service request primitive, the ground-user shall be prohibited from invoking any CPDLC service primitive with this aircraft, except the CPDLC-user-abort request primitive, until after it has received a CPDLC-end service confirmation primitive.

2.3.7.5.4 The DSC-end Service

2.3.7.5.4.1 Receipt of a DSC-end Indication and Invoking DSC-end Response

Note 1.— The CPDLC-ground-user is considered to have downlink open messages when the CPDLC-ground-user has received a message(s) for which a response is required, and it has not yet sent the closure response to the message(s).

Note 2.— Downlink open messages are not considered in setting out the requirements for the DSC-end Service. The air user is aware of any such messages, and desires to end the dialogue anyway. Any such messages are considered deleted upon transmission of a DSC-end response with the Result parameter set to the abstract value “accepted” on the ground side, and upon receipt of a DSC-end confirmation with the Result parameter set to the abstract value “accepted” on the airborne side.

Note 3.— The CPDLC-ground-user is considered to have uplink open messages when the CPDLC-ground-user has sent any messages that require a response for which it has not received a closure response.

Note 4.— Uplink open message are considered deleted upon transmission of a DSC-end response with the Result parameter set to the abstract value “accepted” on the ground side, and upon receipt of a DSC-end confirmation with the Result parameter set to the abstract value “accepted” on the airborne side.

Note 5.— Local procedures may dictate when a "rejected" response Result parameter is permitted by the CPDLC-ground-user in the cases when these SARPs give the CPDLC-ground-user a choice between "accepted" or "rejected".

Note 6.— If a DSC-end-service response is invoked with a "accepted" Result this will result in ending the dialogue regardless of any messages contained in the CPDLC Message parameter.

2.3.7.5.4.1.1 Downlink Message Contains Error

2.3.7.5.4.1.1.1 If a DSC-end indication is received and there is a message in the *CPDLC Message* parameter that either has the response attribute N and requires a logical acknowledgment or has Y response attribute, and an error is detected in the message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the *CPDLC Message* parameter containing a CPDLC message with the message element ERROR [errorInformation],

b) the *Result* parameter set to the abstract value "rejected".

2.3.7.5.4.1.2 No Message in the DSC-end Indication

2.3.7.5.4.1.2.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is no message in the *CPDLC Message* parameter, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted.

2.3.7.5.4.1.2.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is no message in the *CPDLC Message* parameter, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.5.4.1.3 Message in DSC-end Indication Does Not Require Response

2.3.7.5.4.1.3.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted.

2.3.7.5.4.1.3.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.5.4.1.4 Message in DSC-end Indication Requires Only Logical Acknowledgment

2.3.7.5.4.1.4.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the *CPDLC Message* parameter containing a CPDLC message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value “accepted”.

2.3.7.5.4.1.4.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a message in the *CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the *CPDLC Message* parameter containing a CPDLC message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value “accepted” or “rejected.

2.3.7.5.4.1.5 Message in DSC-end Indication With Y Response Attribute

2.3.7.5.4.1.5.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a message in the *CPDLC Message* parameter with the response attribute Y and no error is detected in the message the CPDLC-ground-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element STANDBY,

c) if a REQUEST DEFERRED response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element REQUEST DEFERRED, and

d) invoke DSC-end response with:

1) the *CPDLC Message* parameter containing a Y attribute closure CPDLC message, and

2) the *Result* parameter set to the abstract value “accepted”.

2.3.7.5.4.1.5.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a message in the *CPDLC Message* parameter with the response attribute Y and no error is detected in the message the CPDLC-ground-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element STANDBY, and

c) if a REQUEST DEFERRED response is used, invoke CPDLC-message request with the *CPDLC Message* parameter containing a CPDLC message with at least the message element REQUEST DEFERRED, and

d) invoke DSC-end response with:

1) the *CPDLC Message* parameter containing a Y attribute closure CPDLC message, and

2) the *Result* parameter set to the abstract value “accepted” or “rejected.

2.3.7.5.5 The CPDLC-forward Service

2.3.7.5.5.1 Invoking the CPDLC-forward Request

2.3.7.5.5.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-forward Request.

2.3.7.5.6 The CPDLC-user-abort Service

2.3.7.5.6.1 Issuing a CPDLC-user-abort Request

2.3.7.5.6.1.1 The CPDLC-ground-user shall have the capability to invoke CPDLC-user-abort request with the *Reason* parameter set to CPDLCUserAbortReason value [commanded-termination].

2.3.7.6 Message Intent

2.3.7.6.1 Purpose

Note.— 2.3.7.6 contains the message set for CPDLC. Message attributes, message presentation guidance, and data structure presentation guidance are presented. The actual information exchanged between an aircraft and ground peer or a ground and ground peer CPDLC applications is defined in 2.3.4; however, 2.3.4 does not mandate any particular method for presenting this information. The presentation of information to the controller and aircraft crew is a local implementation. The message presentation recommendations contained in Tables 2.3.7-5 to 2.3.7-28 are one possible means of presenting the information. These recommendations are generally consistent with current ICAO practices for displaying ATC information.

2.3.7.6.2 Uplink message elements shall comply with the intent, use, and element attributes as presented in Tables 2.3.7-5 to 2.3.7-16.

Table 2.3.7-5. Responses/Acknowledgments (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
0	Indicates that ATS cannot comply with the request.	UNABLE	N	M	N
1	Indicates that ATS has received themessage and will respond.	STANDBY	N	L	N
2	Indicates that ATS has received the request but it has been deferred until later.	REQUEST DEFERRED	N	L	N
3	Indicates that ATS has received and understood the message.	ROGER	N	L	N
4	Yes.	AFFIRM	N	L	N
5	No.	NEGATIVE	N	L	N
235	Notification of receipt of unlawful interference message.	ROGER 7500	U	H	N
211	Indicates that the ATS has received the request and has passed it to the Next Control Authority.	REQUEST FORWARDED	N	L	N
218	Indicates to the pilot that the request has already been received on the ground.	REQUEST ALREADY RECEIVED	L	N	N

Table 2.3.7-6. Vertical Clearances (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
6	Notification that a level change instruction should be expected.	EXPECT [level]	L	L	R
7	Notification that an instruction should be expected for the aircraft to commence climb at the specified time.	EXPECT CLIMB AT [time]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
8	Notification that an instruction should be expected for the aircraft to commence climb at the specified position.	EXPECT CLIMB AT [position]	L	L	R
9	Notification that an instruction should be expected for the aircraft to commence descent at the specified time.	EXPECT DESCENT AT [time]	L	L	R
10	Notification that an instruction should be expected for the aircraft to commence descent at the specified position.	EXPECT DESCENT AT [position]	L	L	R
11	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified time.	EXPECT CRUISE CLIMB AT [time]	L	L	R
12	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified position.	EXPECT CRUISE CLIMB AT [position]	L	L	R
13	Notification that an instruction should be expected for the aircraft to commence climb at the specified time to the specified level.	AT [time] EXPECT CLIMB TO [level]	L	L	R
14	Notification that an instruction should be expected for the aircraft to commence climb at the specified position to the specified level.	AT [position] EXPECT CLIMB TO [level]	L	L	R
15	Notification that an instruction should be expected for the aircraft to commence descent at the specified time to the specified level.	AT [time] EXPECT DESCENT TO [level]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
16	Notification that an instruction should be expected for the aircraft to commence descent at the specified position to the specified level.	AT [position] EXPECT DESCENT TO [level]	L	L	R
17	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified time to the specified level.	AT [time] EXPECT CRUISE CLIMB TO [level]	L	L	R
18	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified position to the specified level.	AT [position] EXPECT CRUISE CLIMB TO [level]	L	L	R
19	Instruction to maintain the specified level.	MAINTAIN [level]	N	M	W/U
20	Instruction that a climb to a specified level is to commence and the level is to be maintained when reached.	CLIMB TO [level]	N	M	W/U
21	Instruction that at the specified time, a climb to the specified level is to commence and once reached the specified level is to be maintained.	AT [time] CLIMB TO [level]	N	M	W/U
22	Instruction that at the specified position, a climb to the specified level is to commence and once reached the specified level is to be maintained.	AT [position] CLIMB TO [level]	N	M	W/U
185	Instruction that after passing the specified position, a climb to the specified level is to commence and once reached the specified level is to be maintained.	AFTER PASSING [position] CLIMB TO [level]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
23	Instruction that a descent to a specified level is to commence and the level is to be maintained when reached.	DESCEND TO [level]	N	M	W/U
24	Instruction that at a specified time a descent to a specified level is to commence and once reached the specified level is to be maintained.	AT [time] DESCEND TO [level]	N	M	W/U
25	Instruction that at the specified position a descent to the specified level is to commence and when the specified level is reached it is to be maintained.	AT [position] DESCEND TO [level]	N	M	W/U
186	Instruction that after passing the specified position, a descent to the specified level is to commence and once reached the specified level is to be maintained.	AFTER PASSING [position] DESCEND TO [level]	N	M	W/U
26	Instruction that a climb is to commence at a rate such that the specified level is reached at or before the specified time.	CLIMB TO REACH [level] BY [time]	N	M	W/U
27	Instruction that a climb is to commence at a rate such that the specified level is reached at or before the specified position.	CLIMB TO REACH [level] BY [position]	N	M	W/U
28	Instruction that a descent is to commence at a rate such that the specified level is reached at or before the specified time.	DESCEND TO REACH [level] BY [time]	N	M	W/U
29	Instruction that a descent is to commence at a rate such that the specified level is reached at or before the specified position.	DESCEND TO REACH [level] BY [position]	N	M	W/U
192	Instruction that a change of level is to continue, but at a rate such that the specified level is reached at or before the specified time.	REACH [level] BY [time]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
209	Instruction that a change of level is to continue, but at a rate such that the specified level is reached at or before the specified position	REACH [level] BY [position]	N	M	W/U
30	A level within the defined vertical range specified is to be maintained.	MAINTAI N BLOCK [level] TO [level]	N	M	W/U
31	Instruction that a climb to a level within thevertical range defined is to commence.	CLIMB TO AND MAINTAIN BLOCK [level] TO [level]	N	M	W/U
32	Instruction that a descent to a level within the vertical range defined is to commence.	DESCEND TO AND MAINTAIN BLOCK [level] TO [level]	N	M	W/U
34	A cruise climb is to commence and continue until the specified level is reached.	CRUISE CLIMB TO [level]	N	M	W/U
35	A cruise climb can commence once above the specified level.	CRUISE CLIMB ABOVE [level]	N	M	W/U
219	Instruction to stop the climb below the previously assigned level.	STOP CLIMB AT [level]	U	M	W/U
220	Instruction to stop the descent above the previously assigned level .	STOP DESCENT AT [level]	U	M	W/U
36	The climb to the specified level should be made at the aircraft's best rate.	EXPEDITE CLIMB TO [level]	U	M	W/U
37	The descent to the specified level should be made at the aircraft's best rate.	EXPEDITE DESCENT TO [level]	U	M	W/U
38	Urgent instruction to immediately climb to the specified level.	IMMEDIATELY CLIMB TO [level]	D	H	W/U
39	Urgent instruction to immediately descend to the specified level.	IMMEDIATELY DESCEND TO [level]	D	H	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
40	Urgent instruction to immediately stop a climb once the specified level is reached.	IMMEDIATELY STOP CLIMB AT [level]	D	H	W/U
41	Urgent instruction to immediately stop a descent once the specified level is reached.	IMMEDIATELY STOP DESCENT AT [level]	D	H	W/U
171	Instruction to climb at not less than the specified rate.	CLIMB AT [vertical rate] MINIMUM	N	M	W/U
172	Instruction to climb at not above the specified rate.	CLIMB AT [vertical rate] MAXIMUM	N	M	W/U
173	Instruction to descend at not less than the specified rate.	DESCEND AT [vertical rate] MINIMUM	N	M	W/U
174	Instruction to descend at not above the specified rate.	DESCEND AT [vertical rate] MAXIMUM	N	M	W/U
33	Reserved.				

Table 2.3.7-7. Crossing Constraints (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
42	Notification that a level change instruction should be expected which will require the specified position to be crossed at the specified level.	EXPECT TO CROSS [position] AT [level]	L	L	R
43	Notification that a level change instruction should be expected which will require the specified position to be crossed at or above the specified level.	EXPECT TO CROSS [position] AT OR ABOVE [level]	L	L	R
44	Notification that a level change instruction should be expected which will require the specified position to be crossed at or below the specified level.	EXPECT TO CROSS [position] AT OR BELOW [level]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
45	Notification that a level change instruction should be expected which will require the specified position to be crossed at the specified level which is to be maintained subsequently.	EXPECT TO CROSS [position] AT AND MAINTAIN [level]	L	L	R
46	The specified position is to be crossed at the specified level. This may require the aircraft to modify its climb or descent profile.	CROSS [position] AT [level]	N	M	W/U
47	The specified position is to be crossed at or above the specified level.	CROSS [position] AT OR ABOVE [level]	N	M	W/U
48	The specified position is to be crossed at or below the specified level.	CROSS [position] AT OR BELOW [level]	N	M	W/U
49	Instruction that the specified position is to be crossed at the specified level and that level is to be maintained when reached.	CROSS [position] AT AND MAINTAIN [level]	N	M	W/U
50	The specified position is to be crossed at a level between the specified levels.	CROSS [position] BETWEEN [level] AND [level]	N	M	W/U
51	The specified position is to be crossed at the specified time.	CROSS [position] AT [time]	N	M	W/U
52	The specified position is to be crossed at or before the specified time.	CROSS [position] AT OR BEFORE [time]	N	M	W/U
53	The specified position is to be crossed at or after the specified time.	CROSS [position] AT OR AFTER [time]	N	M	W/U
54	The specified position is to be crossed at a time between the specified times.	CROSS [position] BETWEEN [time] AND [time]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
55	The specified position is to be crossed at the specified speed and the specified speed is to be maintained until further advised.	CROSS [position] AT [speed]	N	M	W/U
56	The specified position is to be crossed at a speed equal to or less than the specified speed and the specified speed or less is to be maintained until further advised.	CROSS [position] AT OR LESS THAN [speed]	N	M	W/U
57	The specified position is to be crossed at a speed equal to or greater than the specified speed and the specified speed or greater is to be maintained until further advised.	CROSS [position] AT OR GREATER THAN [speed]	N	M	W/U
58	The specified position is to be crossed at the specified time and the specified level.	CROSS [position] AT [time] AT [level]	N	M	W/U
59	The specified position is to be crossed at or before the specified time and at the specified level.	CROSS [position] AT OR BEFORE [time] AT [level]	N	M	W/U
60	The specified position is to be crossed at or after the specified time and at the specified level.	CROSS [position] AT OR AFTER [time] AT [level]	N	M	W/U
61	Instruction that the specified position is to be crossed at the specified level and speed and the level and speed are to be maintained.	CROSS [position] AT AND MAINTAIN [level] AT [speed]	N	M	W/U
62	Instruction that at the specified time the specified position is to be crossed at the specified level and the level is to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [level]	N	M	W/U
63	Instruction that at the specified time the specified position is to be crossed at the specified level and speed and the level and speed are to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [level] AT [speed]	N	M	W/U

Table 2.3.7-8. Lateral Offsets (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
64	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction.	OFFSET [specified distance] [direction] OF ROUTE	N	M	W/U
65	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction and commencing at the specified position.	AT [position] OFFSET [specified distance] [direction] OF ROUTE	N	M	W/U
66	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction and commencing at the specified time.	AT [time] OFFSET [specified distance] [direction] OF ROUTE	N	M	W/U
67	The cleared flight route is to be rejoined.	PROCEED BACK ON ROUTE	N	M	W/U
68	The cleared flight route is to be rejoined at or before the specified position.	REJOIN ROUTE BY [position]	N	M	W/U
69	The cleared flight route is to be rejoined at or before the specified time.	REJOIN ROUTE BY [time]	N	M	W/U
70	Notification that a clearance may be issued to enable the aircraft to rejoin the cleared route at or before the specified position.	EXPECT BACK ON ROUTE BY [position]	L	L	R
71	Notification that a clearance may be issued to enable the aircraft to rejoin the cleared route at or before the specified time.	EXPECT BACK ON ROUTE BY [time]	L	L	R
72	Instruction to resume own navigation following a period of tracking or heading clearances. May be used in conjunction with an instruction on how or where to rejoin the cleared route.	RESUME OWN NAVIGATION	N	M	W/U

Table 2.3.7-9. Route Modifications (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
73	Notification to the aircraft of the instructions to be followed from departure until the specified clearance limit.	[departure clearance]	N	M	W/U
74	Instruction to proceed directly from its present position to the specified position.	PROCEED DIRECT TO [position]	N	M	W/U
75	Instruction to proceed, when able, directly to the specified position.	WHEN ABLE PROCEED DIRECT TO [position]	N	M	W/U
76	Instruction to proceed, at the specified time, directly to the specified position.	AT [time] PROCEED DIRECT TO [position]	N	M	W/U
77	Instruction to proceed, at the specified position, directly to the next specified position.	AT [position] PROCEED DIRECT TO [position]	N	M	W/U
78	Instruction to proceed, upon reaching the specified level, directly to the specified position.	AT [level] PROCEED DIRECT TO [position]	N	M	W/U
79	Instruction to proceed to the specified position via the specified route.	CLEARED TO [position] VIA [route clearance]	N	M	W/U
80	Instruction to proceed via the specified route.	CLEARED [route clearance]	N	M	W/U
81	Instruction to proceed in accordance with the specified procedure.	CLEARED [procedure name]	N	M	W/U
236	Instruction to leave controlled airspace.	LEAVE CONTROLLED AIRSPACE	N	M	W/U
82	Approval to deviate up to the specified distance from the cleared route in the specified direction.	CLEARED TO DEVIATE UP TO [specified distance] [direction] OF ROUTE	N	M	W/U
83	Instruction to proceed from the specified position via the specified route.	AT [position] CLEARED [route clearance]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
84	Instruction to proceed from the specified position via the specified procedure.	AT [position] CLEARED [procedure name]	N	M	W/U
85	Notification that a clearance to fly on the specified route may be issued.	EXPECT [route clearance]	L	L	R
86	Notification that a clearance to fly on the specified route from the specified position may be issued.	AT [position] EXPECT [route clearance]	L	L	R
87	Notification that a clearance to fly directly to the specified position may be issued.	EXPECT DIRECT TO [position]	L	L	R
88	Notification that a clearance to fly directly from the first specified position to the next specified position may be issued.	AT [position] EXPECT DIRECT TO [position]	L	L	R
89	Notification that a clearance to fly directly to the specified position commencing at the specified time may be issued.	AT [time] EXPECT DIRECT TO [position]	L	L	R
90	Notification that a clearance to fly directly to the specified position commencing when the specified level is reached may be issued.	AT [level] EXPECT DIRECT TO [position]	L	L	R
91	Instruction to enter a holding pattern with the specified characteristics at the specified position and level.	HOLD AT [position] MAINTAIN [level] INBOUND TRACK [degrees] [direction] TURNS [leg type]	N	M	W/U
92	Instruction to enter a holding pattern with the published characteristics at the specified position and level.	HOLD AT [position] AS PUBLISHED MAINTAIN [level]	N	M	W/U
93	Notification that an onwards clearance may be issued at the specified time.	EXPECT FURTHER CLEARANCE AT [time]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
94	Instruction to turn left or right as specified onto the specified heading.	TURN [direction] HEADING [degrees]	N	M	W/U
95	Instruction to turn left or right as specified onto the specified track.	TURN [direction] GROUND TRACK [degrees]	N	M	W/U
215	Instruction to turn a specified number of degrees left or right.	TURN [direction] [degrees]	N	M	W/U
190	Instruction to fly on the specified heading.	FLY HEADING [degrees]	N	M	W/U
96	Instruction to continue to fly on the current heading.	CONTINUE PRESENT HEADING	N	M	W/U
97	Instruction to fly on the specified heading from the specified position.	AT [position] FLY HEADING [degrees]	N	M	W/U
221	Instruction to stop turn at the specified heading prior to reaching the previously assigned heading.	STOP TURN HEADING [degrees]	U	M	W/U
98	Instruction to turn immediately left or right as specified onto the specified heading.	IMMEDIATELY TURN [direction] HEADING [degrees]	D	H	W/U
99	Notification that a clearance may be issued for the aircraft to fly the specified procedure.	EXPECT [procedure name]	L	L	R

Table 2.3.7-10. Speed Changes (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
100	Notification that a speed instruction may be issued to be effective at the specified time.	AT [time] EXPECT [speed]	L	L	R
101	Notification that a speed instruction may be issued to be effective at the specified position.	AT [position] EXPECT [speed]	L	L	R
102	Notification that a speed instruction may be issued to be effective at the specified level.	AT [level] EXPECT [speed]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
103	Notification that a speed range instruction may be issued to be effective at the specified time.	AT [time] EXPECT [speed] TO [speed]	L	L	R
104	Notification that a speed range instruction may be issued to be effective at the specified position.	AT [position] EXPECT [speed] TO [speed]	L	L	R
105	Notification that a speed range instruction may be issued to be effective at the specified level.	AT [level] EXPECT [speed] TO [speed]	L	L	R
106	The specified speed is to be maintained.	MAINTAIN [speed]	N	M	W/U
188	After passing the specified position the specified speed is to be maintained.	AFTER PASSING [position] MAINTAIN [speed]	N	M	W/U
107	The present speed is to be maintained.	MAINTAIN PRESENT SPEED	N	M	W/U
108	The specified speed or a greater speed is to be maintained.	MAINTAIN [speed] OR GREATER	N	M	W/U
109	The specified speed or a lesser speed is to be maintained.	MAINTAIN [speed] OR LESS	N	M	W/U
110	A speed within the specified range is to be maintained.	MAINTAIN [speed] TO [speed]	N	M	W/U
111	The present speed is to be increased to the specified speed and maintained until further advised.	INCREASE SPEED TO [speed]	N	M	W/U
112	The present speed is to be increased to the specified speed or greater, and maintained at or above the specified speed until further advised.	INCREASE SPEED TO [speed] OR GREATER	N	M	W/U
113	The present speed is to be reduced to the specified speed and maintained until further advised.	REDUCE SPEED TO [speed]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
114	The present speed is to be reduced to the specified speed or less and maintained at or below the specified speed until further advised.	REDUCE SPEED TO [speed] OR LESS	N	M	W/U
115	The specified speed is not to be exceeded.	DO NOT EXCEED [speed]	N	M	W/U
116	Notification that the aircraft need no longer comply with the previously issued speed restriction.	RESUME NORMAL SPEED	N	M	W/U
189	The present speed is to be changed to the specified speed.	ADJUST SPEED TO [speed]	N	M	W/U
222	Instruction that the aircraft may keep its preferred speed without restriction.	NO SPEED RESTRICTION	L	L	R
223	Instruction to reduce present speed to the minimum safe approach speed	REDUCE TO MINIMUM APPROACH SPEED	N	M	W/U

Table 2.3.7-11. Contact/Monitor/Surveillance Requests (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
117	The ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	CONTACT [unitname] [frequency]	N	M	W/U
118	At the specified position the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [position] CONTACT [unitname] [frequency]	N	M	W/U
119	At the specified time the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [time] CONTACT [unitname] [frequency]	N	M	W/U
120	The ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	MONITOR [unitname] [frequency]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
121	At the specified position the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [position] MONITOR [unitname] [frequency]	N	M	W/U
122	At the specified time the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [time] MONITOR [unitname] [frequency]	N	M	W/U
123	The specified code (SSR code) is to be selected.	SQUAWK [code]	N	M	W/U
124	The SSR transponder responses are to be disabled.	STOP SQUAWK	N	M	W/U
125	The SSR transponder responses should include level information.	SQUAWK MODE CHARLIE	N	M	W/U
126	The SSR transponder responses should no longer include level information.	STOP SQUAWK MODE CHARLIE	N	M	W/U
179	The ‘ident’ function on the SSR transponder is to be actuated.	SQUAWK IDENT	N	M	W/U

Table 2.3.7-12. Report/Confirmation Requests (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
127	Instruction to report when the aircraft is back on the cleared route.	REPORT BACK ON ROUTE	N	L	W/U
128	Instruction to report when the aircraft has left the specified level.	REPORT LEAVING [level]	N	L	W/U
129	Instruction to report when the aircraft is in level flight at the specified level.	REPORT MAINTAINING [level]	N	L	W/U
175	Instruction to report when the aircraft has reached the specified level.	REPORT REACHING [level]	N	L	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
180	Instruction to report when the aircraft is within the specified vertical range.	REPORT REACHING BLOCK [level] TO [level]	N	L	W/U
130	Instruction to report when the aircraft has passed the specified position.	REPORT PASSING [position]	N	L	W/U
181	Instruction to report the present distance to or from the specified position.	REPORT DISTANCE [to/from] [position]	N	M	Y
184	Instruction to report at the specified time the distance to or from the specified position.	AT TIME [time] REPORT DISTANCE [to/from] [position]	N	L	Y
228	Instruction to report the estimated time of arrival at the specified position	REPORT ETA [position]	L	L	Y
131	Instruction to report the amount of fuel remaining and the number of persons on board.	REPORT REMAINING FUEL AND PERSONS ON BOARD	U	M	Y
132	Instruction to report the present position.	REPORT POSITION	N	M	Y
133	Instruction to report the present level.	REPORT PRESENT LEVEL	N	M	Y
134	Instruction to report the requested speed.	REPORT [speed type] [speed type] [speed type] SPEED	N	M	Y
135	Instruction to confirm and acknowledge the currently assigned level.	CONFIRM ASSIGNED LEVEL	N	L	Y
136	Instruction to confirm and acknowledge the currently assigned speed.	CONFIRM ASSIGNED SPEED	N	L	Y
137	Instruction to confirm and acknowledge the currently assigned route.	CONFIRM ASSIGNED ROUTE	N	L	Y
138	Instruction to confirm the previously reported time over the last reported waypoint.	CONFIRM TIME OVER REPORTED WAYPOINT	N	L	Y

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
139	Instruction to confirm the identity of the previously reported waypoint.	CONFIRM REPORTED WAYPOINT	N	L	Y
140	Instruction to confirm the identity of the next waypoint.	CONFIRM NEXT WAYPOINT	N	L	Y
141	Instruction to confirm the previously reported estimated time at the next waypoint.	CONFIRM NEXT WAYPOINT ETA	N	L	Y
142	Instruction to confirm the identity of the next but one waypoint.	CONFIRM ENSUING WAYPOINT	N	L	Y
143	The request was not understood. It should be clarified and resubmitted.	CONFIRM REQUEST	N	L	Y
144	Instruction to report the selected code (SSR).	CONFIRM SQUAWK	N	L	Y
145	Instruction to report the present heading.	REPORT HEADING	N	M	Y
146	Instruction to report the present ground track.	REPORT GROUND TRACK	N	M	Y
182	Instruction to report the identification code of the last ATIS received.	CONFIRM ATIS CODE	N	L	Y
147	Instruction to make a position report.	REQUEST POSITION REPORT	N	M	Y
216	Instruction to file a flight plan.	REQUEST FLIGHT PLAN	N	M	Y
217	Instruction to report that the aircraft has landed.	REPORT ARRIVAL	N	M	Y
229	Instruction to report the preferred alternate aerodrome for landing.	REPORT ALTERNATE AERODROME	L	L	Y
231	Instruction to indicate the pilot's preferred level.	STATE PREFERRED LEVEL	L	L	Y
232	Instruction to indicate the pilot's preferred time and/or position to commence descent to the aerodrome of intended arrival.	STATE TOP OF DESCENT	L	L	Y

Table 2.3.7-13. Negotiation Requests (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
148	Request for the earliest time at which the specified level can be accepted.	WHEN CAN YOU ACCEPT [level]	N	L	Y
149	Instruction to report whether or not the specified level can be accepted at the specified position.	CAN YOU ACCEPT [level] AT [position]	N	L	A/N
150	Instruction to report whether or not the specified level can be accepted at the specified time.	CAN YOU ACCEPT [level] AT [time]	N	L	A/N
151	Instruction to report the earliest time when the specified speed can be accepted.	WHEN CAN YOU ACCEPT [speed]	N	L	Y
152	Instruction to report the earliest time when the specified offset track can be accepted.	WHEN CAN YOU ACCEPT [specified distance] [direction] OFFSET	N	L	Y

Table 2.3.7-14. Air Traffic Advisories (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
153	ATS advisory that the altimeter setting should be the specified setting.	ALTIMETER [altimeter]	N	L	R
213	ATS advisory that the specified altimeter setting relates to the specified facility.	[facility designation] ALTIMETER [altimeter]	N	L	R
154	ATS advisory that the radar service is terminated.	RADAR SERVICE TERMINATED	N	L	R
191	ATS advisory that the aircraft is entering airspace in which no air traffic services are provided and all existing air traffic services are terminated.	ALL ATS TERMINATED	N	M	R
155	ATS advisory that radar contact has been established at the specified position.	RADAR CONTACT [position]	N	M	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
156	ATS advisory that radar contact has been lost.	RADAR CONTACT LOST	N	M	R
210	ATS advisory that the aircraft has been identified on radar at the specified position.	IDENTIFIED [position]	N	M	R
193	Indication that radar identification has been lost.	IDENTIFICATION LOST	N	M	R
157	A continuous transmission is detected on the specified frequency. Check the microphone button.	CHECK STUCK MICROPHONE [frequency]	U	M	N
158	ATS advisory that the ATIS information identified by the specified code is the current ATIS information.	ATIS [atis code]	N	L	R
212	ATS advisory that the specified ATIS information at the specified airport is current.	[facility designation] ATIS [atis code] CURRENT	N	L	R
214	ATS advisory that indicates the RVR value for the specified runway.	RVR RUNWAY [runway] [rvr]	N	M	R
224	ATS advisory that no delay is expected.	NO DELAY EXPECTED	N	L	R
225	ATS advisory that the expected delay has not been determined.	DELAY NOT DETERMINED	N	L	R
226	ATS advisory that the aircraft may expect to be cleared to commence its approach procedure at the specified time.	EXPECTED APPROACH TIME [time]	N	L	R

Table 2.3.7-15. System Management Messages (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
159	A system generated message notifying that the ground system has detected an error.	ERROR [error information]	U	M	N
160	Notification to the avionics that the next data authority is the specified ATSU.	NEXT DATA AUTHORITY [facility]	L	N	N
161	Notification to the avionics that the data link connection with the current data authority is being terminated.	END SERVICE	L	N	N
162	Notification that the ground system does not support this message.	SERVICE UNAVAILABLE	L	L	N
234	Notification that the ground system does not have a flight plan for that aircraft.	FLIGHT PLAN NOT HELD	L	L	N
163	Notification to the pilot of an ATSU identifier.	[facility designation]	L	N	N
227	Confirmation to the aircraft system that the ground system has received the message to which the logical acknowledgment refers and found it acceptable for display to the responsible person.	LOGICAL ACKNOWLEDGMENT	N	M	N
233	Notification to the pilot that messages sent requiring a logical acknowledgment will not be accepted by this ground system.	USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED	N	M	N

Table 2.3.7-16. Additional Messages (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
164	The associated instruction may be complied with at any future time.	WHEN READY	L	N	N
230	The associated instruction is to be complied with immediately.	IMMEDIATELY	D	H	N
165	Used to link two messages, indicating the proper order of execution of clearances/ instructions.	THEN	L	N	N
166	The associated instruction is issued due to traffic considerations.	DUE TO [traffic type] TRAFFIC	L	N	N
167	The associated instruction is issued due to airspace restrictions.	DUE TO AIRSPACE RESTRICTION	L	N	N
168	The indicated communication should be ignored.	DISREGARD	U	M	R
176	Notification that the operator is responsible for maintaining separation from other traffic and is also responsible for maintaining Visual Meteorological Conditions.	MAINTAIN OWN SEPARATION AND VMC	N	M	W/U
177	Used in conjunction with a clearance/instruction to indicate that the operator may execute when prepared to do so.	AT PILOTS DISCRETION	L	L	N
169		[free text]	N	L	R
170		[free text]	D	H	R
194		[free text]	N	L	Y
178		[free text]	N	L	N
195		[free text]	L	L	R
196		[free text]	N	M	W/U
197		[free text]	U	M	W/U
198		[free text]	D	H	W/U
199		[free text]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
200	Used in conjunction with a level clearance to report reaching the level assigned.	REPORT REACHING	N	M	R
201	*Not Used.*		L	L	N
202		[free text]	D	H	W/U
203		[free text]	N	M	R
204		[free text]	N	M	Y
183		[free text]	N	M	N
205		[free text]	N	M	A/N
206		[free text]	L	N	Y
187		[free text]	L	N	N
207		[free text]	L	L	Y
208		[free text]	L	L	N

2.3.7.6.3 Downlink message elements shall comply with the intent, use, and element attributes as presented in Tables 2.3.7-17 to 2.3.7-28.

Table 2.3.7-17. Responses (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
0	The instruction is understtod and will be complied with.	WILCO	N	M	N
1	The instruction cannot be complied with.	UNABLE	N	M	N
2	Wait for a reply.	STANDBY	N	M	N
3	Message received and understood.	ROGER	N	M	N
4	Yes.	AFFIRM	N	M	N
5	No.	NEGATIVE	N	M	N

Table 2.3.7-18. Vertical Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Request to fly at the specified level.	REQUEST [level]	N	L	Y
7	Request to fly at a level within the specified vertical range.	REQUEST BLOCK [level] TO [level]	N	L	Y
8	Request to cruise climb to the specified level.	REQUEST CRUISE CLIMB TO [level]	N	L	Y
9	Request to climb to the specified level.	REQUEST CLIMB TO [level]	N	L	Y
10	Request to descend to the specified level.	REQUEST DESCENT TO [level]	N	L	Y
11	Request that at the specified position a climb to the specified level be approved.	AT [position] REQUEST CLIMB TO [level]	N	L	Y
12	Request that at the specified position a descent to the specified level be approved.	AT [position] REQUEST DESCENT TO [level]	N	L	Y
13	Request that at the specified time a climb to the specified level be approved.	AT [time] REQUEST CLIMB TO [level]	N	L	Y
14	Request that at the specified time a descent to the specified level be approved.	AT [time] REQUEST DESCENT TO [level]	N	L	Y
69	Request that a descent be approved on a see-and-avoid basis.	REQUEST VMC DESCENT	N	L	Y

Table 2.3.7-19. Lateral Off-Set Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
15	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved.	REQUEST OFFSET [specified distance] [direction] OF ROUTE	N	L	Y

	Message Intent/Use	Message Element	URG	ALRT	RESP
16	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved from the specified position.	AT [position] REQUEST OFFSET [specified distance] [direction] OF ROUTE	N	L	Y
17	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved from the specified time.	AT [time] REQUEST OFFSET [specified distance] [direction] OF ROUTE	N	L	Y

Table 2.3.7-20. Speed Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
18	Request to fly at the specified speed.	REQUEST [speed]	N	L	Y
19	Request to fly within the specified speed range.	REQUEST [speed] TO [speed]	N	L	Y

Table 2.3.7-21. Voice Contact Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
20	Request for voice contact.	REQUEST VOICE CONTACT	N	L	Y
21	Request for voice contact on the specified frequency.	REQUEST VOICE CONTACT [frequency]	N	L	Y

Table 2.3.7-22. Route Modification Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
22	Request to track from the present position direct to the specified position.	REQUEST DIRECT TO [position]	N	L	Y
23	Request for the specified procedure clearance.	REQUEST [procedure name]	N	L	Y
24	Request for a route clearance.	REQUEST CLEARANCE [route clearance]	N	L	Y
25	Request for a clearance.	REQUEST [clearance type] CLEARANCE	N	L	Y
26	Request for a weather deviation to the specified position via the specified route.	REQUEST WEATHER DEVIATION TO [position] VIA [route clearance]	N	M	Y
27	Request for a weather deviation up to the specified distance off track in the specified direction.	REQUEST WEATHER DEVIATION UP TO [specified distance] [direction] OF ROUTE	N	M	Y
70	Request a clearance to adopt the specified heading.	REQUEST HEADING [degrees]	N	L	Y
71	Request a clearance to adopt the specified ground track.	REQUEST GROUND TRACK [degrees]	N	L	Y

Table 2.3.7-23. Reports (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
28	Notification of leaving the specified level.	LEAVING [level]	N	L	N
29	Notification of climbing to the specified level.	CLIMBING TO [level]	N	L	N
30	Notification of descending to the specified level.	DESCENDING TO [level]	N	L	N
31	Notification of passing the specified position.	PASSING [position]	N	L	N
78	At the specified time, the aircraft's position was as specified.	AT [time] [distance] [to/from] [position]	N	L	N

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
32	Notification of the present level.	PRESENT LEVEL [level]	N	L	N
33	Notification of the present position.	PRESENT POSITION [position]	N	L	N
34	Notification of the present speed.	PRESENT SPEED [speed]	N	L	N
113	Notification of the requested speed .	[speed type] [speed type] [speed type] SPEED [speed]	N	L	N
35	Notification of the present heading in degrees.	PRESENT HEADING [degrees]	N	L	N
36	Notification of the present ground track in degrees.	PRESENT GROUND TRACK [degrees]	N	L	N
37	Notification that the aircraft is maintaining the specified level.	LEVEL [level]	N	L	N
72	Notification that the aircraft has reached the specified level.	REACHING [level]	N	L	N
76	Notification that the aircraft has reached a level within the specified vertical range.	REACHING BLOCK [level] TO [level]	N	L	N
38	Read-back of the assigned level.	ASSIGNED LEVEL [level]	N	M	N
77	Read-back of the assigned vertical range.	ASSIGNED BLOCK [level] TO [level]	N	M	N
39	Read-back of the assigned speed.	ASSIGNED SPEED [speed]	N	M	N
40	Read-back of the assigned route.	ASSIGNED ROUTE [route clearance]	N	M	N
41	The aircraft has regained the cleared route.	BACK ON ROUTE	N	M	N
42	The next waypoint is the specified position.	NEXT WAYPOINT [position]	N	L	N
43	the ETA at the next waypoint is as specified.	NEXT WAYPOINT ETA [time]	N	L	N
44	The next but one waypoint is the specified position.	ENSUING WAYPOINT [position]	N	L	N
45	Clarification of previously reported waypoint passage.	REPORTED WAYPOINT [position]	N	L	N

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
46	Clarification of time over previously reported waypoint.	REPORTED WAYPOINT [time]	N	L	N
47	The specified code has been selected.	SQUAWKING [code]	N	L	N
48	Position report.	POSITION REPORT [position report]	N	M	N
79	The code of the latest ATIS received is as specified.	ATIS [atis code]	N	L	N
89	The specified ICAO unit is being monitored on the specified frequency.	MONITORING [unitname] [frequency]	U	M	N
102	Used to report that an aircraft has landed.	LANDING REPORT	N	N	N
104	Notification of estimated time of arrival at the specified position.	ETA [position][time]	L	L	N
105	Notification of the alternative aerodrome for landing.	ALTERNATE AERODROME [airport]	L	L	N
106	Notification of the preferred level.	PREFERRED LEVEL [level]	L	L	N
109	Notification of the preferred time to commence descent for approach.	TOP OF DESCENT [time]	L	L	N
110	Notification of the preferred position to commence descent for approach.	TOP OF DESCENT [position]	L	L	N
111	Notification of the preferred time and position to commence descent for approach.	TOP OF DESCENT [time] position]	L	L	N

Table 2.3.7-24. Negotiation Requests (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
49	Request for the earliest time at which a clearance to the specified speed can be expected.	WHEN CAN WE EXPECT [speed]	L	L	Y

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
50	Request for the earliest time at which a clearance to a speed within the specified range can be expected.	WHEN CAN WE EXPECT [speed] TO [speed]	L	L	Y
51	Request for the earliest time at which a clearance to regain the planned route can be expected.	WHEN CAN WE EXPECT BACK ON ROUTE	L	L	Y
52	Request for the earliest time at which a clearance to descend can be expected.	WHEN CAN WE EXPECT LOWER LEVEL	L	L	Y
53	Request for the earliest time at which a clearance to climb can be expected.	WHEN CAN WE EXPECT HIGHER LEVEL	L	L	Y
54	Request for the earliest time at which a clearance to cruise climb to the specified level can be expected.	WHEN CAN WE EXPECT CRUISE CLIMB TO [level]	L	L	Y
87	Request for the earliest time at which a clearance to climb to the specified level can be expected.	WHEN CAN WE EXPECT CLIMB TO [level]	L	L	Y
88	Request for the earliest time at which a clearance to descend to the specified level can be expected.	WHEN CAN WE EXPECT DESCENT TO [level]	L	L	Y

Table 2.3.7-25. Emergency Messages (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
55	Urgency prefix.	PAN PAN PAN	U	H	Y
56	Distress prefix.	MAYDAY MAYDAY MAYDAY	D	H	Y
112	Indicates specifically that the aircraft is being subjected to unlawful interference.	SQUAWKING 7500	U	H	N
57	Notification of fuel remaining and number of persons on board.	[remaining fuel] OF FUEL REMAINING AND [persons on board] PERSONS ON BOARD	U	H	N

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
58	Notification that the pilot wishes to cancel the emergency condition.	CANCEL EMERGENCY	U	M	Y
59	Notification that the aircraft is diverting to the specified position via the specified route.	DIVERTING TO [position] VIA [route clearance]	U	H	Y
60	Notification that the aircraft is deviating the specified distance in the specified direction off the cleared route and maintaining a parallel track.	OFFSETTING [specified distance] [direction] OF ROUTE	U	H	Y
61	Notification that the aircraft is descending to the specified level.	DESCENDING TO [level]	U	H	Y
80	Notification that the aircraft is deviating from the cleared route by the specified distance in the specified direction.	DEVIATING [specified distance] [direction] OFF ROUTE	U	H	Y

Table 2.3.7-26. System Management Messages (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
62	A system generated message that the avionics has detected an error.	ERROR [error information]	U	L	N
63	A system generated denial to any CPDLC message sent from a ground facility that is not the Current Data Authority.	NOT CURRENT DATA AUTHORITY	L	L	N
99	A system generated message to inform a ground facility that it is now the Current Data Authority	CURRENT DATA AUTHORITY	L	L	N
107	A system generated message sent to a ground system that tries to connect to an aircraft when a current data authority has not designated the ground system as the NDA.	NOT AUTHORIZED NEXT DATA AUTHORITY	L	L	N

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
64	Notification to the ground system that the specified ATSU is the current data authority.	[facility designation]	L	L	N
73	A system generated message indicating the software version number.	[version number]	L	L	N
100	Notification to the ground system that the aircraft system has received the message to which the logical acknowledgment refers.	LOGICAL ACKNOWLEDGMENT	N	M	N

Table 2.3.7-27. Additional Messages (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
65	Used to explain reasons for aircraft operator's message.	DUE TO WEATHER	L	L	N
66	Used to explain reasons for aircraft operator's message.	DUE TO AIRCRAFT PERFORMANCE	L	L	N
74	States a desire by the aircraft operator to provide his/her own separation and remain in VMC.	REQUEST TO MAINTAIN OWN SEPARATION AND VMC	L	L	Y
75	Used in conjunction with another message to indicate that the operator wishes to execute request when the pilot is prepared to do so.	AT PILOTS DISCRETION	L	L	N
101	Allows the aircraft operator to indicate a desire for termination of CPDLC service with the current data authority.	REQUEST END OF SERVICE	L	L	Y
103	Allows the aircraft operator to indicate that he/she has canceled IFR flight plan.	CANCELLING IFR	N	L	Y
108	Notification that de-icing action has been completed.	DE-ICING COMPLETE	L	L	N
67		[free text]	N	L	N
68		[free text]	D	H	Y

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
90		[free text]	N	M	N
91		[free text]	N	L	Y
92		[free text]	L	L	Y
93		[free text]	U	H	N
94		[free text]	D	H	N
95		[free text]	U	M	N
96		[free text]	U	L	N
97		[free text]	L	L	N
98		[free text]	N	N	N

Table 2.3.7-28. Negotiation Responses (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
81	We can accept the specified level at the specified time.	WE CAN ACCEPT [level] AT [time]	L	L	N
82	We cannot accept the specified level.	WE CANNOT ACCEPT [level]	L	L	N
83	We can accept the specified speed at the specified time.	WE CAN ACCEPT [speed] AT [time]	L	L	N
84	We cannot accept the specified speed.	WE CANNOT ACCEPT [speed]	L	L	N
85	We can accept a parallel track offset the specified distance in the specified direction at the specified time.	WE CAN ACCEPT [specified distance] [direction] at [time]	L	L	N
86	We cannot accept a parallel track offset the specified distance in the specified direction.	WE CANNOT ACCEPT [specified distance] [direction]	L	L	N

2.3.7.7 Parameter Value Unit, Range, and Resolution

2.3.7.7.1 A CPDLC-user shall interpret CPDLC message element variables unit, range, and resolution as defined in 2.3.4.

2.3.8 SUBSETTING RULES

2.3.8.1 General

Note.— This chapter specifies conformance requirements which all implementations of the CPDLC protocol obey.

2.3.8.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service claiming conformance to 2.3 shall support the CPDLC protocol features as shown in the tables below:

Note.— The 'status' column indicates the level of support required for conformance to the CPDLC-ASE protocol described in this part. The values are as follows:

'M' mandatory support is required,

'O' optional support is permitted for conformance to the CPDLC protocol,

'N/A' the item is not applicable, and

'C.n' the item is conditional where n is the number which identifies the condition which is applicable. The definitions for the conditional statements used in this chapter are written under the tables in which they first appear.

Table 2.3. 8-1. Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 2.3.8-2. CPDLC Protocol Options

	Status	**Associated Predicate**
CPDLC-air-ASE	C.1	CPDLC/air
CPDLC-ground-ASE	C.1	CPDLC/ground
DSC function supported	if (CPDLC/air) O, else M	DSC-FU
DSC function supported by CPDLC-ground-user	if (CPDLC/ground) O, else N/A	DSC-USER
Forward function supported by initiating user	if (CPDLC/ground) O, else N/A	FWD-INIT
Forward function supported by receiving user	if (CPDLC/ground) O, else N/A	FWD-USER

C.1: a conforming implementation will support one and only one of these two options.

Table 2.3.8-3. CPDLC-air-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I.	CPDLC/air	a CPDLC-air-ASE supporting just the core* CPDLC functionality (no downstream clearance capability)
II.	CPDLC/air + DSC-FU	a CPDLC-air-ASE supporting the core CPDLC functionality and the downstream clearance capability (complete CPDLC-air-ASE functionality)
		* the core CPDLC functionality is defined as support for the CPDLC-start, message, end services plus abort services.

Table 2.3.8-4. CPDLC-ground-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I.	CPDLC/ground	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for receiving and indicating that the forward function is not supported
II.	CPDLC/ground + DCS-FU + DSC-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for receiving and indicating that the ground forward function is not supported
III.	CPDLC/ground + DSC-FU + FWD-INIT	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for receiving and indicating that the ground forward function is not supported • functionality for supporting the capability to initiate the ground forwarding of CPDLC messages
IV.	CPDLC/ground + DCS-FU + DSC-USER + FWD-INIT	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for receiving and indicating that the ground forward function is not supported • functionality for supporting the capability to initiate the ground forwarding of CPDLC messages

	List of Predicates	**Functionality Description**
V.	CPDLC/ground + DSC-FU FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for DSC dialogue.
VI.	CPDLC/ground + DCS-FU + DSC-USER + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for CPDLC-ground-user support for the receipt of CPDLC ground forward messages
VII.	CPDLC/ground + DSC-FU + FWD-INIT + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • full CPDLC ground forwarding functionality (initiating and user receiving)
VIII.	CPDLC/ground + DSC-FU + DSC-USER + FWD-INIT + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • full CPDLC ground forwarding functionality (initiating and user receiving) (complete CPDLC-ground-ASE functionality)

Table 2.3.8-5. Supported CPDLC Service Primitives

	Sending (req,[cnf])	Receiving (ind, [rsp])
CPDLC-start service	M	M
CPDLC-message service	M	M
CPDLC-end service	if (CPDLC/ground) M, else N/A	if (CPDLC/air) M, else N/A
DSC-start service	if (CPDLC/air and DSC-FU) M, else N/A	if (CPDLC/ground) M, else N/A

	Sending (req,[cnf])	Receiving (ind, [rsp])
DSC-end service	if (CPDLC/air and DSC-FU) M, else N/A	if (CPDLC/ground and DSC-USER) M, else N/A
CPDLC-forward service	if (FWD-INIT) M, else N/A	if (FWD-USER) M, else N/A
CPDLC-user-abort	M	M
CPDLC-provider-abort	N/A	M

Table 2.3.8-6. Supported CPDLC APDUs

	Sender	**Receiver**
[GroundPDUs] startup	if (CPDLC/ground) M, else N/A	if (CPDLC/air) M, else N/A
[GroundPDUs] send	if (CPDLC/ground) M, else N/A	if (CPDLC/air) M, else N/A
[GroundPDUs] forward	if (FWD-INIT) M, else N/A	if (CPDLC/ground) M, else N/A
[GroundPDUs] forwardresponse	if (CPDLC/ground) M, else N/A	if (FWD-INIT) M, else N/A
[GroundPDUs] abort	if (CPDLC/ground) M, else N/A	if (CPDLC/air or FWD-INIT) M, else N/A
[AircraftPDUs] startdown	if (CPDLC/air) M, else N/A	if (CPDLC/ground) M, else N/A
[AircraftPDUs] send	if (CPDLC/air) M, else N/A	if (CPDLC/ground) M, else N/A
[AircraftPDUs] abort	if (CPDLC/air) M, else N/A	if (CPDLC/ground) M, else N/A

2.4 FLIGHT INFORMATION SERVICES APPLICATION

2.4.1 INTRODUCTION

2.4.1.1 Introduction

2.4.1.1.1 The FIS application allows a pilot to request and receive FIS services from ground FIS systems. The FIS application is designed to enable FIS services to be provided to a pilot via the exchange of messages between aircraft avionics and ground FIS systems.

Note.— Structure:

a) 2.4.1: INTRODUCTION contains the part's purpose and structure and a summary of the functions of FIS.

b) 2.4.2: GENERAL REQUIREMENTS contains backwards compatibility and error processing requirements.

c) 2.4.3: ABSTRACT SERVICE contains the description of the abstract service provided by the application service elements (ASE) defined for FIS.

d) 2.4.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of the messages exchanged by FIS-ASEs using the Abstract Syntax Notation Number One (ASN.1).

e) 2.4.5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the FIS protocol, as well as time constraints and the exception handling procedures associated with these exchanges. This chapter also describes the FIS protocol in terms of state tables.

f) 2.4.6: COMMUNICATION REQUIREMENTS contains the requirements that the FIS-ASEs impose on the underlying communication system.

g) 2.4.7: FIS USER REQUIREMENTS outlines the requirements that a user of a FIS-ASE must meet.

h) 2.4.8: SUBSETTING RULES contains the conformance requirements which all implementations of the FIS protocol obey.

2.4.1.1.2 Two types of FIS contract may be established on request of the pilot:

a) the *FIS Demand Contract* where the ground FIS system provides the information immediately and once only, and

b) the *FIS Update Contract* where the ground FIS system provides the information and any subsequent update of this information.

Note.— The concept of the FIS Demand Contract and the FIS Update Contract used in 2.4 is equivalent to the concept of FIS Demand Mode and FIS Contract Mode developed in the ICAO Manual of Air Traffic Services (ATS) Data Link Applications.

2.4.1.1.3 Multiple "FIS services" may be supported by the FIS application, as for instance:

a) Automatic Terminal Information Services (ATIS),

b) Precipitation Map Service,

c) Terminal Weather Service (TWS),

d) Windshear Advisory Service,

e) Pilot Report (PIREP) Service,

f) Notice to Airmen (NOTAM) Service, and

g) Runway Visual Range (RVR) Service.

2.4.1.1.4 Each of these services will be accessed and used independently of the others and are initiated by the aircraft avionics (and/or pilot). It will not be required that aircraft avionics include the capability for all of the FIS services.

2.4.1.1.5 The FIS application supports only the FIS service related to ATIS. Additional services and negotiation mechanisms could be incorporated in future packages.

Note.— Functional Descriptions

*a) The **FIS Demand Contract** function:*

1) This function allows the airborne FIS system to establish a FIS demand contract with a ground FIS system. Realisation of the contract involves the sending of a single FIS report from the ground FIS system to the aircraft, optionally after the sending of a positive acknowledgement.

2) Multiple FIS demand contracts may be established in parallel with a ground FIS server.

3) The actions performed by the FIS systems supporting the FIS Demand Contract function are the following:

i) the airborne FIS system formats and sends to the ground FIS system a FIS-demand-contract message. This message identifies the type of FIS information requested and contains the parameters of the request, and

ii) the ground FIS system then determines whether or not it is able to comply with the request:

A) if the ground FIS system detects that the requested FIS information can be retrieved but is not yet available, the ground FIS system formats and sends to the airborne FIS system a FIS-positive-acknowledgement message first to indicate its acceptance of the contract, and the FIS-report message later,

B) if the ground FIS system can comply promptly with the FIS demand contract request it sends the FIS-report message as soon as possible, or

C) if there are errors in the FIS-demand-contract message, or if the ground FIS system cannot comply with the request, it sends a FIS-contract-reject message to the airborne FIS system indicating the reason for its inability to accept the contract.

4) When an unrecoverable error situation is detected by the airborne or the ground FIS system or when either of the users request the abrupt termination of the FIS demand contract, the FIS system formats and sends to the peer system a FIS-abort message indicating the source and the reason of the abort. All FIS contracts established between the ground FIS system and the airborne FIS system are aborted.

*5) The **FIS-demand-contract** message contains the following information:*

i) the reference of the FIS contract,

ii) the type of FIS information requested, and

iii) the parameters of the ATIS request, i.e.:

A) the airport identifier, and

B) optionally, the type of the requested ATIS (arrival or departure).

*6) The **FIS-report** message contains the following information:*

i) the reference of the FIS contract,

ii) the type of the FIS information returned, and

iii) the parameters of the ATIS, i.e.:

A) the airport identifier,

B) the version number of the returned ATIS,

C) the type of the returned ATIS (departure, arrival, both or combined),

D) optionally, the time of observation of the returned ATIS, and

E) the ATIS information elements, i.e.:

I) the mandatory ATIS elements: identification of the runway(s) in use, runway surface conditions, other operational information, surface winds, visual visibility, cloud, air temperature, dew point temperature, altimeter setting, SIGMET, specific ATIS instructions, and

II) the optional ATIS elements: approach type, braking action, holding delay, transition level, runway visibility range, present weather, trend type landing forecast.

*7) The **FIS-positive-acknowledgement** message contains the reference of the FIS contract.*

*8) The **FIS-contract-reject** message contains the following information:*

i) the reference of the FIS contract, and

ii) the reason of the rejection.

*9) The **FIS-abort** message contains the following information:*

i) the type of the FIS contract aborted,

ii) the source of the abort of the FIS contract (i.e. FIS service-provider or FIS service-user), and

iii) if the source is the FIS service-provider, the reason of the abort.

*b) The **FIS Update Contract** function:*

1) This function allows the airborne FIS system to establish an Update Contract with a ground FIS system. Realisation of the contract involves the sending of FIS reports from the ground FIS system to the aircraft each time the requested FIS information is modified.

2) Multiple FIS update contracts may be established in parallel with a ground FIS server.

3) The actions performed by the FIS systems supporting the FIS Update Contract function are the following:

i) the airborne FIS system formats and sends to the ground FIS system a FIS-update-contract message. This message identifies the type of FIS information requested and contains the parameters of the request, and

ii) if the ground FIS system can comply with the FIS-update-contract request, then

A) it sends the first FIS-report message, as soon as possible, and

B) whenever the requested FIS information is modified, it sends a new FIS-report message to the aircraft.

iii) if the ground FIS system detects that the requested FIS information can be retrieved but is not yet available, then

A) it formats and sends to the airborne FIS system a FIS-positive-acknowledgement message first to indicate its acceptance of the contract, and

B) then sends the FIS-report messages,

iv) if there are errors in the FIS-update-contract message, or if the ground FIS system cannot comply with the request, it sends a FIS-contract-reject message to the airborne FIS system indicating the reason for its inability to accept the contract, or

v) if the ground FIS system does not support the update contract function, it sends a FIS-contract-reject message containing, if available, the requested FIS information.

4) When an error situation is detected by the airborne or the ground FIS system or when either of the users request the abrupt termination of the FIS

Update Contract, the FIS system formats and sends to the peer system a FIS-abort message indicating the source and the reason of the abort. All FIS contracts established between the ground FIS system and the airborne FIS system are aborted.

*5) The **FIS-update-contract** message has the same contents as the FIS-demand-contract message as described for the FIS Demand Contract function.*

*6) The **FIS-report** messages have the same contents as the FIS-report message as described for the FIS Demand Contract function.*

*7) The **FIS-contract-reject** message contains the following information:*

i) the reference of the FIS contract,

ii) the reason of the rejection of the FIS contract, and

iii) the current value of the requested ATIS, if the reason of the rejection is "FIS Update Contract function not supported by the ground FIS system".

*8) The **FIS-positive-acknowledgement** message has the same contents as the FIS-positive-acknowlegement message as described for the FIS Demand Contract function.*

*9) The **FIS-abort** message has the same contents as the FIS-abort message as described for the FIS Demand Contract function.*

*c) The **Cancellation of Contracts** function:*

1) This function allows both air and ground FIS system to cancel a particular FIS update contract that is in operation, as follows:

i) a FIS-update-contract-cancel message is sent by the system initiating the termination. Any FIS-report message previously sent is delivered to the aircraft before the contract is effectively ended. Other pending FIS contracts are not disrupted by the termination of a particular FIS update contract, and

ii) the cancellation of the FIS update contract is confirmed to the FIS-user by the system receiving the FIS-update-contract-cancel message. A FIS-update-contract-cancel-accept message is sent back.

2) The airborne FIS system may also cancel all FIS contracts (demand and update contracts) of the same type in a single FIS-cancel-contracts message. The ground FIS system cancels these contracts and acknowledges the cancellation by sending a FIS-cancel-contracts-accept message. The cancellation is made on the basis of the type(s) of contract supplied by the airborne FIS system.

*3) The **FIS-update-contract-cancel** message contains the following information:*

i) the reference of the FIS contract, and

ii) the type of the FIS-update-contract.

*4) The **FIS-update-contract-cancel-accept** message contains the following information:*

i) the reference of the FIS contract, and

ii) the type of the FIS-update-contract cancelled.

*5) The **FIS-cancel-contracts** message contains the types of the FIS contracts to be cancelled.*

*6) The **FIS-cancel-contracts-accept** message contains the types of the cancelled FIS contracts.*

2.4.2 GENERAL REQUIREMENTS

2.4.2.1 FIS ASE Version Number

Note.— 2.4 describes the version 1 of the protocol operated by the FIS-ASEs. Best efforts will be made to ensure that subsequent versions of this protocol are backwards compatible.

2.4.2.1.1 For this version of the FIS SARPs, the FIS-air-ASE and FIS-ground-ASE version numbers shall both be set to one.

2.4.2.2 Error Processing Requirements

2.4.2.2.1 In the event of information input by the FIS-user being incompatible with that able to be processed by the system, the FIS-user shall be notified.

2.4.2.2.2 In the event of a FIS-user invoking a FIS service primitive when the FIS-ASE is not in a state specified in 2.4.5, the FIS-user shall be notified.

2.4.3 THE ABSTRACT SERVICE

2.4.3.1 Service Description

2.4.3.1.1 An implementation of either the FIS ground based service or the FIS air based service shall exhibit external behaviour consistent with having implemented a FIS-ground-ASE, or a FIS-air-ASE respectively.

Note 1.— 2.4.3 defines the abstract service interface for the FIS service. The FIS-ASE abstract service is described in this chapter from the viewpoint of the FIS-air-user, the FIS-ground-user and the FIS service-provider.

Note 2.— 2.4.3 defines the static behaviour (i.e, the format) of the FIS-ASE abstract service. Its dynamic behaviour (i.e., how it is used) is described in 2.4.7.

Note 3.— Figure 2.4.3-1 shows the functional model of the FIS Application. The functional modules identified in this model are the following:

a) the FIS-user,

b) the FIS Application Entity (FIS-AE) service interface,

c) the FIS-AE,

d) the FIS Control Function (FIS-CF),

e) the FIS Application Service Element (FIS-ASE) service interface,

f) the FIS-ASE, and

g) the Dialogue Service (DS) interface.

Note 4.— The FIS-user represents the operational part of the FIS system. This user does not perform the communication functions but relies on a communication service provided to it via the FIS-AE through the FIS-AE service interface. The individual actions at this interface are called FIS-AE service primitives. Similarly, individual actions at other interfaces in the communication system are called service primitives at these interfaces.

Note 5.— The FIS-AE consists of several elements, including the FIS-ASE and the FIS-CF. The DS interface is made available by the FIS-CF to the FIS-ASE for communication with the peer FIS-ASE.

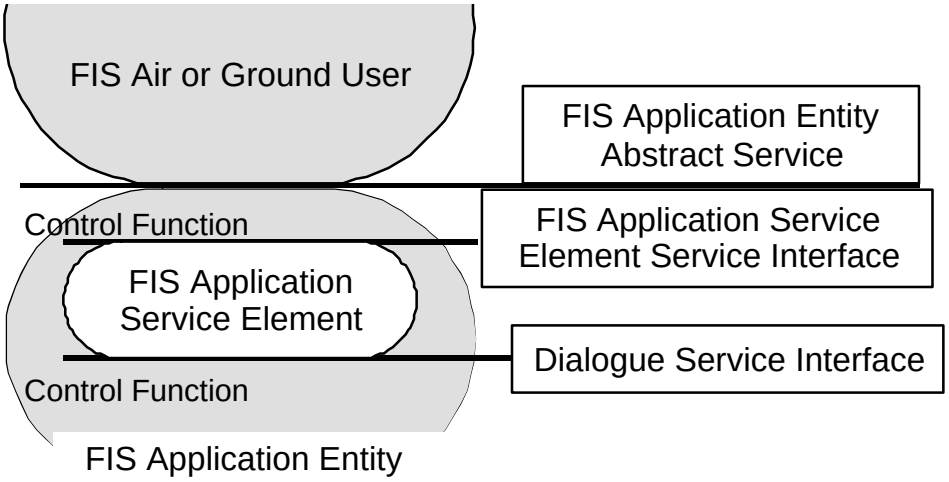

Figure 2.4.3-1. Functional Model of the FIS Application

Note 6.— The FIS-ASE is the element in the communication system which executes the FIS specific protocol. In other words, it takes care of the FIS specific service primitive sequencing actions, message creation, timer management, error and exception handling.

Note 7.— The FIS-ASE interfaces only with the FIS-CF. This FIS-CF is responsible for mapping service primitives received from one element (such as the FIS-ASE and the FIS-user) to other elements which interface with it. The part of the FIS-CF which is relevant from the point of view of these SARPs, i.e. the part between the FIS-user and the FIS-ASE, will map FIS-AE service primitives to FIS-ASE service primitives transparently.

Note 8.— The DS interface is the interface between the FIS-ASE and part of FIS-CF underneath the FIS-ASE, and provides the dialogue service as described in 4.2.

2.4.3.2 The FIS-ASE Abstract Service

Note.— There is no requirement to implement the service in a FIS product; however, it is necessary to implement the ground based and air based system in such a way that it will be impossible to detect (from the peer system) whether or not an interface has been built.

2.4.3.2.1 The FIS-ASE abstract service shall consist of a set of the following services as allowed by the subsetting rules defined in 2.4.8:

a) *FIS-demand-contract* service as defined in 2.4.3.3,

b) *FIS-update-contract* service as defined in 2.4.3.4,

c) *FIS-report* service as defined in 2.4.3.5,

d) *FIS-cancel-contracts* service as defined in 2.4.3.6,

e) *FIS-cancel-update-contract* service as defined in 2.4.3.7,

f) *FIS-user-abort* service as defined in 2.4.3.8, and

g) *FIS-provider-abort* service as defined in 2.4.3.9.

Note 1.— For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables of this chapter:

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, and*
f)	***U***	*user option.*

Note 2.— The following abbreviations are used in this part:

a)	***Req***	*request; data is input by FIS-user initiating the service to its respective ASE,*
b)	***Ind***	*indication; data is indicated by the receiving ASE to its respective FIS-user,*
c)	***Rsp***	*response; data is input by receiving FIS-user to its respective ASE, and*
d)	***Cnf***	*confirmation; data is confirmed by the initiating ASE to its respective FIS-user.*

Note 3.— An unconfirmed service allows just one message to be transmitted, in one direction, without providing a corresponding response.

Note 4.— A confirmed service provides end-to-end confirmation that a message sent by one user was received by its peer user.

Note 5.— An abstract syntax is a syntactical description of a parameter which does not imply a specific implementation. Only when the FIS-ASE maps a parameter onto an APDU field, or vice-versa, is the abstract syntax of the parameter described by using the ASN.1 of 2.4.4 for this field.

2.4.3.3 FIS-demand-contract Service

Note.— The FIS-demand-contract service allows the FIS-air-user to request a FIS demand contract with a FIS-ground-user. It is a confirmed service, initiated by the FIS-air-user.

2.4.3.3.1 The FIS-demand-contract service shall contain primitives and parameters as presented in Table 2.4.3-1.

Table 2.4.3-1. FIS-demand-contract service parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
ICAO Facility Designation	M			
FIS Contract Number	M	M(=)	M(=)	M(=)
Class of Communication Service	U			
FIS Contract Details	M	M(=)		
Result			M	M(=)
Reject Reason			C	C(=)
FIS Information			C	C(=)

2.4.3.3.2 ICAO Facility Designation

Note.— This parameter contains the FIS-ground-ASE ICAO Facility Designation.

2.4.3.3.2.1 The *ICAOFacilityDesignation* parameter value shall conform to the abstract syntax four to eight-character *ICAO Facility Designation*.

2.4.3.3.3 FIS Contract Number

Note.— This parameter contains the user-defined reference of the requested FIS demand contract.

2.4.3.3.3.1 The *FISContractNumber* parameter value shall conform to the ASN.1 abstract syntax *ContractNumber*.

2.4.3.3.4 Class Of Communication Service

Note.— This parameter contains the value of the required class of communication service, if specified by the FIS-air-user.

2.4.3.3.4.1 Where specified by the FIS-air-user, the *ClassOfCommunicationService* parameter value shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G” or “H”.

Note 1.— If FIS contracts are currently in place, the ClassOfCommunicationService parameter is not used by the FIS service provider.

Note 2.— Where not specified by the FIS-air-user, when there are no FIS Contracts already in force, this indicates that there is no routing preference.

2.4.3.3.5 FIS Contract Details

Note.— This parameter contains the details of the FIS Demand Contract as requested by the FIS-air-user.

2.4.3.3.5.1 The *FISContractDetails* parameter value shall conform to the ASN.1 abstract syntax *FISRequestData*.

Note.— This parameter identifies also the type of the requested FIS information. For version 1 of the FIS-ASEs, the requested information is always of type “ATIS”.

2.4.3.3.6 Result

Note.— The value of this parameter indicates whether the FIS-demand-contract request has been accepted or rejected by the FIS-ground-user.

2.4.3.3.6.1 The *Result* parameter value shall conform to one of the following abstract values:

a) “accepted”,

b) “positive acknowledgment”, and

c) “rejected”.

2.4.3.3.7 Reject Reason

Note.— This parameter contains the reason of the rejection.

2.4.3.3.7.1 The *RejectReason* parameter value shall conform to the ASN.1 abstract syntax *FISRejectReason.*

2.4.3.3.7.2 The *RejectReason* parameter shall be present if and only if the *Result* parameter contains the abstract value “rejected”.

2.4.3.3.8 FIS Information

Note.— This parameter contains the FIS information, as requested by the FIS-air-user.

2.4.3.3.8.1 The *FISInformation* parameter value shall conform to the ASN.1 abstract syntax *FISReportData*.

Note.— This parameter identifies also the type of the returned FIS information. For version 1 of the FIS-ASEs, the requested information is always of type "ATIS".

2.4.3.3.8.2 The *FISInformation* parameter shall be present if and only if the *Result* parameter contains the abstract value "accepted".

2.4.3.4 FIS-update-contract Service

Note.— The FIS-update-contract service allows the FIS-air-user to request a FIS update contract with a FIS-ground-user. It is a confirmed service, initiated by the FIS-air-user.

2.4.3.4.1 The FIS-update-contract service shall contain primitives and parameters as presented in Table 2.4.3-2.

Table 2.4.3-2. FIS-update-contract service parameters

Parameter Name	Req	Ind	Rsp	Cnf
ICAO Facility Designation	M			
FIS Contract Number	M	M(=)	M(=)	M(=)
Class Of Communication Service	U			
FIS Contract Details	M	M(=)		
Result			M	M(=)
Reject Reason			C	C(=)
FIS Information			C	C(=)

2.4.3.4.2 ICAO Facility Designation

Note.— This parameter contains the FIS-ground-ASE ICAO Facility Designation.

2.4.3.4.2.1 The *ICAOFacilityDesignation* parameter value shall conform to the abstract syntax four to eight-character *ICAO Facility Designation*.

2.4.3.4.3 FIS Contract Number

Note.— This parameter contains the user-defined reference of the requested FIS update contract.

2.4.3.4.3.1 The *FISContractNumber* parameter value shall conform to the ASN.1 abstract syntax *ContractNumber*.

2.4.3.4.4 Class Of Communication Service

Note.— This parameter contains the value of the required class of communication service, if specified by the FIS-air-user.

2.4.3.4.4.1 Where specified by the FIS-air-user, the *ClassOfCommunicationService* parameter value shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G” or “H”.

Note 1.— If FIS contracts are currently in place, the ClassOfCommunicationService parameter is not used by the FIS service provider.

Note 2.— Where not specified by the FIS-air-user, when there are no FIS Contracts already in force, this indicates that there is no routing preference.

2.4.3.4.5 FIS Contract Details

Note.— This parameter contains the details of the FIS Update Contract as requested by the FIS-air-user.

2.4.3.4.5.1 The *FISContractDetails* parameter value shall conform to the ASN.1 abstract syntax *FISRequestData*.

Note.— This parameter identifies also the type of the requested FIS information. For version 1 of the FIS-ASEs, the requested information is always of type “ATIS”.

2.4.3.4.6 Result

Note.— The value of this parameter indicates whether the FIS-update-contract request has been accepted or rejected by the FIS-ground-user.

2.4.3.4.6.1 The *Result* parameter value shall conform to one of the following abstract values:

a) “accepted”,

b) “positive acknowledgment”, and

c) “rejected”.

2.4.3.4.7 Reject Reason

Note.— This parameter contains the reason of the rejection.

2.4.3.4.7.1 The *RejectReason* parameter value shall conform to one of the following abstract values:

a) “can not comply”,

b) “FIS service unavailable”,

c) “error detected in the FIS request”,

d) “update contract function not supported by the FIS-ground-user”, and

e) “undefined”.

2.4.3.4.7.2 The *RejectReason* parameter shall be present if and only if the *Result* parameter contains the abstract value “rejected”.

2.4.3.4.8 FIS Information

Note.— This parameter contains the FIS information requested by the FIS-air-user.

2.4.3.4.8.1 The *FISInformation* parameter value shall conform to the ASN.1 abstract syntax *FISReportData*.

Note.— This parameter identifies also the type of the returned FIS information. For version 1 of the FIS-ASEs, the requested information is always of type “ATIS”.

2.4.3.4.8.2 The *FISInformation* parameter shall be present if the *Result* parameter contains the abstract value “accepted”.

Note.— The FISInformation parameter is present upon user’s choice if the Result parameter contains the abstract value “rejected” and the RejectReason parameter contains the abstract value “update contract function not supported by the FIS-ground-user”.

2.4.3.5 FIS-report Service

Note.— The FIS-report service allows the FIS-ground-user to send to the FIS-air-user the requested FIS information and any update of this FIS information. It is an unconfirmed service initiated by the FIS-ground-user.

2.4.3.5.1 The FIS-report service shall contain primitives and parameters as presented in Table 2.4.3-3.

Table 2.4.3-3. FIS-report service parameters

Parameter Name	Req	Ind
FIS Contract Number	M	M(=)
FIS Information	M	M(=)

2.4.3.5.2 FIS Contract Number

Note.— This parameter contains the user-defined reference of the FIS demand contract the FIS Information is related to.

2.4.3.5.2.1 The *FISContractNumber* parameter value shall conform to the ASN.1 abstract syntax *ContractNumber*.

2.4.3.5.3 FIS Information

Note.— This parameter contains the FIS information requested by the FIS-air-user.

2.4.3.5.3.1 The *FISInformation* parameter value shall conform to the ASN.1 abstract syntax *FISReportData*.

Note.— This parameter identifies also the type of the returned FIS information. For version 1 of the FIS-ASEs, the requested information is always of type "ATIS".

2.4.3.6 FIS-cancel-contracts Service

Note.— The FIS-cancel-contracts service allows the FIS-air-user to cancel all FIS demand and update contracts of the same type with a particular FIS-ground-user. It is a confirmed service initiated by the FIS-air-user.

2.4.3.6.1 The FIS-cancel-contracts service shall contain primitives and parameters as presented in Table 2.4.3-4.

Table 2.4.3-4. FIS-cancel-contracts service parameters

Parameter Name	Req	Ind	Cnf
FIS Service Type	M	M(=)	M(=)

2.4.3.6.2 FIS Service Type

Note.— This parameter identifies the types of the FIS contracts to be cancelled.

2.4.3.6.2.1 The *FISServiceType* parameter value shall conform to the ASN.1 abstract syntax *FISCancelContracts*.

Note.— For version 1 of the FIS-ASEs, this parameter will always identify the service type "ATIS".

2.4.3.7 FIS-cancel-update-contract Service

Note.— The FIS-cancel-update-contract service allows the FIS-air-user or the FIS-ground-user to cancel in an orderly manner an active FIS update contract. The FIS reports in transit in the communication system are delivered before the FIS update contract is cancelled. This service does not affect the other FIS contracts. This is a confirmed service, initiated by the FIS-air-user or the FIS-ground-user.

2.4.3.7.1 The FIS-cancel-update-contract service shall contain primitives and parameters as presented in Table 2.4.3-5.

Table 2.4.3-5. FIS-cancel-update-contract service parameters

Parameter Name	Req	Ind	Cnf
FIS Contract Number	M	M(=)	M(=)

2.4.3.7.2 FIS Contract Number

Note.— This parameter contains the reference of the FIS update contract to be cancelled.

2.4.3.7.2.1 The *FISContractNumber* parameter value shall conform to the ASN.1 abstract syntax *ContractNumber*.

2.4.3.8 FIS-user-abort Service

Note 1.— The FIS-user-abort service allows a FIS-user to abort all active FIS contracts (both FIS demand contracts and FIS update contracts). As a consequence, all active FIS contracts processed by the ASE are cancelled. Messages in transit may be lost during this operation. This is an unconfirmed service. It can be invoked at any time that the FIS-user is aware that any FIS service is in operation.

Note 2.— If the FIS-user-abort service is invoked prior the complete establishment of the dialogue, the FIS-user-abort indication may not be provided. A FIS-provider-abort-indication may result instead.

2.4.3.8.1 The FIS-user-abort service shall contain the primitives as presented in Table 2.4.3-6.

Table 2.4.3-6. FIS-user-abort service parameters

Parameter Name	Req	Ind
none		

2.4.3.9 FIS-provider-abort Service

Note.— The FIS-provider-abort service is used by the FIS service-provider to inform its active users that it can no longer provide the FIS service. As a consequence, all active FIS contracts (both FIS demand contract and FIS update contract) are cancelled. This service is a FIS service-provider initiated service. Messages in transit may be lost during this operation.

2.4.3.9.1 The FIS-provider-abort service shall contain primitives and parameters as presented in Table 2.4.3-7.

Table 2.4.3-7. FIS-provider-abort service parameters

Parameter Name	Ind
Reason	M

2.4.3.9.2 Reason

Note.— This parameter contains the reason for the abort.

2.4.3.9.2.1 The *Reason* parameter value shall conform to the ASN.1 abstract syntax *FISProtocolErrorDiag*.

2.4.4 FORMAL DEFINITIONS OF MESSAGES

2.4.4.1 Encoding/Decoding Rules

2.4.4.1.1 A FIS-air-ASE shall be capable of encoding [FISDownlinkAPDU] APDUs and decoding [FISUplinkAPDU] APDUs.

2.4.4.1.2 A FIS-ground-ASE shall be capable of encoding [FISUplinkAPDU] APDUs and decoding [FISDownlinkAPDU] APDUs.

2.4.4.2 FIS ASN.1 Abstract Syntax

2.4.4.2.1 The abstract syntax of the FIS protocol data units shall comply with the description contained in the ASN.1 module FISMessageSetVersion1 (conforming to ISO/IEC 8824-1), as defined in 2.4.4.

```
 FISMessageSetVersion1 DEFINITIONS::=

BEGIN
-- ------------------------------------------------------------------------
-- FIS messages (Top level)
-- ------------------------------------------------------------------------
FISDownlinkAPDU ::= SEQUENCE
        {
        time                    DateTimeGroup,
        fisDownlinkAPDU         DownlinkAPDU
        }

FISUplinkAPDU ::= SEQUENCE
        {
        time                    DateTimeGroup,
        fisUplinkAPDU           UplinkAPDU
        }

DownlinkAPDU::= CHOICE
        {
        fISRequest              [0] FISRequest,
        fISCancelUpdateContract [1] FISCancelUpdateContract,
        fISCancelUpdateAccept [2] FISCancelUpdateAccept,
        fISCancelContracts      [3] FISCancelContracts,
        fISAbort                [4] FISAbort,
        ...
        }
```

```
UplinkAPDU::= CHOICE
        {
        fISAccept                   [0] FISAccept,
        fISReject                   [1] FISReject,
        fISReport                   [2] FISReport,
        fISCancelUpdateContract     [3] FISCancelUpdateContract,
        fISCancelUpdateAccept [4] FISCancelUpdateAccept,
        fISCancelContractsAccept    [5] FISCancelContractsAccept,
        fISAbort                    [6] FISAbort,
        ...
        }

-- ------------------------------------------------------------------------
-- FIS messages (2nd level)
-- ------------------------------------------------------------------------
FISAbort ::= CHOICE
        {
        -- Automatic Terminal Information Service (ATIS)
        atis                        [0] FISProtocolErrorDiag,
        ...
        }

FISAccept  ::= SEQUENCE
        {
        contractNumber              ContractNumber,
        fISAcceptData               FISAcceptData
        }

FISAcceptData::= CHOICE
        {
        accept                      [0] FISReportData,
        positiveAcknowledgement     [1] NULL
        }

FISCancelContracts ::=              FISServiceType

FISCancelContractsAccept ::=        FISServiceType
```

```
FISCancelAcceptData::= CHOICE
        {
        -- Automatic Terminal Information Service (ATIS)
        atis                                    [0] NULL,
        ...
        }

FISCancelUpdateAccept ::= SEQUENCE
        {
        fISUpdateContractNumber                 ContractNumber,
        fISCancelAcceptData                     FISCancelAcceptData
        }

FISCancelUpdateContract ::= SEQUENCE
        {
        fISUpdateContractNumber                 ContractNumber,
        fISCancelUpdateData                     FISCancelUpdateData
        }

FISCancelUpdateData ::= CHOICE
        {
        -- Automatic Terminal Information Service (ATIS)
        atis                                    [0] NULL,
        ...
        }

FISReject ::= SEQUENCE
        {
        contractNumber                          ContractNumber,
        fISRejectData                           FISRejectData
        }

FISRejectData ::= CHOICE
        {
        updateFunctionNotSupported              [0] NULL,
        updateFunctionNotSupportedWithReport    [1] FISReportData,
        otherReasons                            [2] FISRejectReason,
        ...
        }
```

```
FISRejectReason ::= ENUMERATED
        {
        canNotComply                (0),
        fISServiceUnavailable       (1),
        errorInRequest              (2),
        undefined                   (3),
        ...
        }

FISReport ::= SEQUENCE
        {
        contractNumber              ContractNumber,
        fISReportData               FISReportData
        }

FISReportData ::= CHOICE
        {
        -- Automatic Terminal Information Service (ATIS)
        atis                        [0] ATISReport,
        ...
        }

FISRequest ::= SEQUENCE
        {
        contractNumber              ContractNumber,
        contractType                ContractType            DEFAULT demandContract,
        fISRequestData              FISRequestData
        }

FISRequestData ::= CHOICE
        {
        -- Automatic Terminal Information Service (ATIS)
        aTISRequest                 [0] ATISRequest,
        ...
        }
```

```
FISProtocolErrorDiag ::= ENUMERATED
        {
        timerExpiration          (0),
        protocolError            (1),
        sequenceError            (2),
        decodingError            (3),
        unrecoverableInternalError (4),
        invalidContractNumber    (5),
        dialogueEndNotSupported  (6),
        undefined                (7),
        invalidQosParameter      (8),
        ...
        }

FISServiceType ::= BIT STRING
        {
        -- Automatic Terminal Information Service (ATIS)
        atis                     (0)
        }
        SIZE (1,...)

-- ------------------------------------------------------------------------------------------------
-- ATIS Messages
-- ------------------------------------------------------------------------------------------------
 ATISInformation ::= CHOICE
        {
        arrivalATIS              [0] ArrivalATIS,
        departureATIS            [1] DepartureATIS,
        combinedATIS             [2] CombinedATIS,
        arrivalAndDepartureATIS  [3] ArrivalAndDepartureATIS
        }

ATISReport ::= SEQUENCE
        {
        aerodromeID              Aerodrome,
        aTISInformation          ATISInformation
        }

ATISRequest ::= SEQUENCE
        {
        aerodromeID              Aerodrome,
        arrivalDepartureIndicator ArrivalDepartureIndicator  DEFAULT arrival
        }
```

```
ArrivalAndDepartureATIS ::= SEQUENCE
        {
        arrivalATIS             ArrivalATIS,
        departureATIS           DepartureATIS
        }

ArrivalATIS ::= SEQUENCE
        {
        aTISDesignator          [0] ATISDesignator,
        aTISTimeOfObservation   [1] Time                        OPTIONAL,
        arrivalRunwaysInUse     [2] SEQUENCE (SIZE (1..36))
                                        OF ArrivalRunway,
        commonATISInfo          [3] CommonATISInformation,
        arrivalATISInfo         [4] SpecificATISArrivalInfo
        }

CombinedATIS ::= SEQUENCE
        {
        aTISDesignator          [0] ATISDesignator,
        aTISTimeOfObservation   [1] Time                        OPTIONAL,
        runwaysInUse            [2] SEQUENCE (SIZE (1..36))
                                        OF RunwayType,
        commonATISInfo          [3] CommonATISInformation,
        arrivalATISInfo         [4] SpecificATISArrivalInfo
        }

DepartureATIS ::= SEQUENCE
        {
        aTISDesignator          [0] ATISDesignator,
        aTISTimeOfObservation   [1] Time                        OPTIONAL,
        departureRunwaysInUse   [2] SEQUENCE (SIZE (1..36))
                                        OF DepartureRunway,
        commonATISInfo          [3] CommonATISInformation
        }

CommonATISInformation ::= SEQUENCE
        {
        surfaceWind             [0] SurfaceWind,
        visibility              [1] Visibility                  OPTIONAL,
        cloud                   [2] Cloud,
        airTemperature          [3] Temperature,
        dewPointTemperature     [4] Temperature,
        altimeterSetting        [5] AltimeterSetting,
        presentWeather          [6] PresentWeather              OPTIONAL,
        significantMetPhenomena [7] SignificantMetPhenomena     OPTIONAL,
        holdingDelay            [8] IA5String (SIZE(1..200))    OPTIONAL,
```

```
        specificATISInstructions        [9] IA5String (SIZE (1..64))        OPTIONAL,
        otherOperationInfo              [10] IA5String (SIZE (1..250))      OPTIONAL,
        transitionLevel                 [11] TransitionLevel                OPTIONAL,
        ...
        }

SpecificATISArrivalInfo ::= SEQUENCE
        {
        trendTypeLandingForecast        IA5String (SIZE (1..256))           OPTIONAL,
        ...
        }

-- ------------------------------------------------------------------------------------------------
-- ATIS fields
-- Note: this should be read in conjunction with ICAO Annexes 3, 4, 5, 11, 14 and 15
-- ------------------------------------------------------------------------------------------------

Aerodrome ::= IA5String (SIZE(4))

AltimeterSetting ::= SEQUENCE
        {
        qNH                             [0] PressureMeasure,
        qFEAerodrome                    [1] PressureMeasure                 OPTIONAL,
        qFERunway                       [2] SEQUENCE (SIZE(1..36))
                                            OF RunwayQFE                    OPTIONAL
        }

Approach ::= ENUMERATED
        {
        ils                             (0),
        localizer                       (1),
        ndb                             (2),
        vor                             (3),
        vordme                          (4),
        nonprecisiongps                 (5),
        precisiongps                    (6),
        dmearc                          (7),
        precisionapproachradar          (8),
        asr                             (9),
        visual                          (10),
        rnav                            (11),
        chartedvisualapproachprocedure  (12),
        lda                             (13),
        fms                             (14),
        loran                           (15),
        mls                             (16),
```

```
        ilsdme                  (17),
        localizerbackcourse     (18),
        localizerDME            (19),
        vortac                  (20),
        tacan                   (21),
        ndbDme                  (22),
        vdf                     (23),
        sra                     (24),
        ...
        }

ApproachType ::= SEQUENCE
        {
        approachType            [0] Approach,
        approachTypeOther       [1] IA5String (SIZE (1..30))    OPTIONAL
        }

ArrivalDepartureIndicator ::= ENUMERATED
        {
        arrival                 (0),
        departure               (1),
        combined                (2)
        }

ArrivalRunway ::= SEQUENCE
        {
        runway                  [0] Runway,
        approachType            [1] ApproachType                OPTIONAL,
        circleRunway            [2] RunwayId                    OPTIONAL
        }

ATISDesignator ::= IA5String (SIZE(1))

BrakingAction ::= SEQUENCE
        {
        brakingActionFull       [0] BrakingActionDescription    OPTIONAL,
        brakingActionFirstThird [1] BrakingActionDescription    OPTIONAL,
        brakingActionSecondThird [2] BrakingActionDescription   OPTIONAL,
        brakingActionLastThird  [3] BrakingActionDescription    OPTIONAL
        }

BrakingActionDescription ::= SEQUENCE
        {
        brakingActionQuality    BrakingActionQuality,
        brakingActionQualifier  IA5String (SIZE (1..25))
        }
```

```
BrakingActionQuality ::= ENUMERATED
        {
        good                    (0),
        mediumToGood            (1),
        medium                  (2),
        mediumToPoor            (3),
        poor                    (4)
        }

Cloud ::= CHOICE
        {
        cloudInfo               [0] CloudInformation,
        skyObscured             [1] SkyObscured,
        cAVOK                   [2] NULL
        }

CloudAmount ::= ENUMERATED
        {
        skyclear                (0),
        few                     (1),
        scattered               (2),
        broken                  (3),
        overcast                (4)
        }

CloudHeight ::= CHOICE
        {
        lowCloudHeightMeters    [0] INTEGER (0..100),
        -- Units = meters, range (0..3000), resolution = 30
        lowCloudHeightFeet      [1] INTEGER (0..100),
        -- Units = feet, range (0..10000), resolution = 100
        highCloudHeightMeters [2] INTEGER (11..67),
        -- Units = meters, range (3300..20100), resolution = 300
        highCloudHeightFeet     [3] INTEGER (11..60)
        -- Units = feet, range (11000..60000), resolution = 1000
}

CloudInformation ::= SEQUENCE
        {
        cloudAmount             [0] CloudAmount,
        cloudHeight             [1] CloudHeight         OPTIONAL,
        cloudType               [2] CloudType           OPTIONAL
        }
```

```
CloudType ::= ENUMERATED
        {
        cumulonimbus            (0),
        toweringCumulus         (1)
        }

CombinedRunway ::= SEQUENCE
        {
        runway                  [0] Runway,
        approachType            [1] ApproachType        OPTIONAL,
        circleRunway            [2] RunwayId            OPTIONAL
        }

ContractNumber ::= INTEGER (1..256)

ContractType ::= ENUMERATED
        {
        demandContract          (0),
        updateContract          (1)
        }

Date ::= SEQUENCE
        {
        year                    Year,
        month                   Month,
        day                     Day
        }

DateTimeGroup ::= SEQUENCE
        {
        date                    Date,
        time                    HHMMSS
        }

Day ::= INTEGER (1..31)
-- unit = day, range (1..31), resolution = 1

DepartureRunway ::= Runway

DescriptorQualifier ::= ENUMERATED
        {
        shallow                 (0),
        partial                 (1),
        patches                 (2),
        lowDrifting             (3),
        blowing                 (4),
```

```
        shower                  (5),
        thunderstorm            (6),
        freezing                (7)
        }

HHMMSS ::= SEQUENCE
        {
        timeHours               TimeHours,
        timeMinutes             TimeMinutes,
        timeSeconds             TimeSeconds
        }

IntensityQualifier ::= ENUMERATED
        {
        light                   (0),
        moderate                (1),
        heavy                   (2)
        }

Month ::= INTEGER (1..12)
-- unit = month, range (1..12), resolution = 1

Obscuration ::= ENUMERATED
        {
        mist                    (0),
        fog                     (1),
        smoke                   (2),
        volcanicAsh             (3),
        widespreadDust          (4),
        sand                    (5),
        haze                    (6)
        }

OtherWeatherPhenomena ::= ENUMERATED
        {
        dustSandWhirls          (0),
        squalls                 (1),
        funnelCloudTornadoWaterspout (2),
        sandstorm               (3),
        duststorm               (4)
        }

Precipitation ::= ENUMERATED
        {
        drizzle                 (0),
        rain                    (1),
```

```
        snow                        (2),
        snowGrains                  (3),
        iceCrystals                 (4),
        icePellets                  (5),
        hail                        (6),
        smallHailAndOrSnowPellets   (7),
        unknownPrecipitation        (8)
        }

PresentWeather ::= SEQUENCE
        {
        presentWeather      [0] PresentWX,
        nosig                   [1] NULL                    OPTIONAL
        }

PresentWX ::= SEQUENCE
        {
        type                    [0] PresentWeatherType,
        intensityQualifier      [1] IntensityQualifier      OPTIONAL,
        inTheVicinity           [2] NULL                    OPTIONAL,
        descriptorQualifier     [3] DescriptorQualifier     OPTIONAL
        }

PresentWeatherType ::= SEQUENCE
        {
        precipitation           [0] Precipitation           OPTIONAL,
        obscuration             [1] Obscuration             OPTIONAL,
        otherWeatherPhenomena   [2] OtherWeatherPhenomena   OPTIONAL
        }

PressureMeasure ::= CHOICE
        {
        hPa                     [0] INTEGER (500..1250),
        -- units = hPa, range (500..1250), resolution = 1
        inches                  [1] INTEGER (2200..3200)
        -- units = inches of Mercury, range (22.00..32.00), resolution= 0.01
        }
```

```
Runway ::= SEQUENCE
        {
        runwayId                    [0] RunwayId,
        runwaySurfaceConditions     [1] RunwaySurfaceConditions     OPTIONAL,
        brakingAction               [2] BrakingAction               OPTIONAL,
        runwayArrestingSystem [3] SEQUENCE (SIZE (1..2))
                                        OF RunwayArrestingSystem    OPTIONAL,
        runwayVisualRange           [4] RVR                         OPTIONAL
        }

RASCondition ::= ENUMERATED
        {
        up                          (0),
        down                        (1),
        unavailable                 (2)
        }

RASLocation ::= ENUMERATED
        {
        arrivalEnd                  (0),
        departureEnd                (1)
        }

RunwayArrestingSystem ::= SEQUENCE
        {
        location                    RASLocation,
        condition                   RASCondition
        }

RunwayDesignator  ::= INTEGER (1..36)

RunwayId ::= SEQUENCE
        {
        runwayDesignator            [0] RunwayDesignator,
        runwayLetter                [1] RunwayLetter                OPTIONAL
        }

RunwayLetter  ::= ENUMERATED
        {
        leftLeft                    (0),
        left                        (1),
        leftCenter                  (2),
        center                      (3),
        rightCenter                 (4),
        right                       (5),
        rightRight                  (6)
```

```
        }

RunwayQFE ::= SEQUENCE
        {
        runwayId                RunwayId,
        qFE                     PressureMeasure
        }

RunwaySurfaceConditions ::= SEQUENCE
        {
        surfaceConditions       [0] SurfaceConditions,
        other                   [1] IA5String (SIZE (1..256))       OPTIONAL
        }

RunwayType  ::= CHOICE
        {
        arrivalRunway           [0] ArrivalRunway,
        departureRunway         [1] DepartureRunway,
        combinedRunway          [2] CombinedRunway
        }

RunwayVisibility ::= CHOICE
        {
        inoperative             [0] NULL,
        reported                [1] RVRVisibility
        }

RVR ::= SEQUENCE
        {
        touchdownRVR            [0] RunwayVisibility,
        midPointRVR             [1] RunwayVisibility                OPTIONAL,
        stopEndRVR              [2] RunwayVisibility                OPTIONAL,
        rVRVariationQualifier   [3] SEQUENCE (SIZE(2))
                                        OF RVRVisibility            OPTIONAL
        }
```

```
RVRVisibility ::= CHOICE
        {
        lowVisibilityMeters             [0] INTEGER (0..32),
        -- units = meters, range (0..800), resolution = 25
        highVisibilityMeters            [1] INTEGER (9..15),
        -- units = meters, range (900..1500), resolution = 100
        lowVisibilityFeet               [2] INTEGER (0..10),
        -- units = feet, range (0..1000), resolution = 100
        midVisibilityFeet               [3] INTEGER (6..15),
        -- units = feet, range (1200..3000), resolution = 200
        highVisibilityFeet              [4] INTEGER (7..12)
        -- units = feet, range (3500..6000), resolution = 500
        }

SkyObscured ::= CHOICE
        {
        noVerticalVisibility            [0] NULL,
        verticalVisibility              [1] VerticalVisibility
        }

SignificantMetPhenomena ::= SEQUENCE
        {
        approachAreaMet             [0] IA5String (SIZE (1..128))       OPTIONAL,
        takeoffAreaMet          [1] IA5String (SIZE (1..128))       OPTIONAL,
        climboutAreaMet             [2] IA5String (SIZE (1..128))       OPTIONAL
        }

SurfaceConditions ::= ENUMERATED
        {
        damp                            (0),
        wet                             (1),
        waterPatches                    (2),
        flooded                         (3),
        wetSnow                         (4),
        drySnow                         (5),
        snow                            (6),
        slush                           (7),
        ice                             (8)
        }

SurfaceWind ::= CHOICE
        {
        calmIndicator                   [0] NULL,
        surfaceWind                     [1] SurfaceWD
        }
```

```
SurfaceWD ::= SEQUENCE
        {
        surfaceWindDirection        [0] SurfaceWindDirection,
        surfaceWindSpeed            [1] SurfaceWindSpeed,
        surfaceWindVariations       [2] SurfaceWindVariations       OPTIONAL
        }

SurfaceWindDirection ::= INTEGER (1..36)
-- units = degree, range (10..360), resolution = 10
-- wind direction is the direction from which the wind is coming

SurfaceWindSpeed ::= CHOICE
        {
        windSpeedMeters             [0] INTEGER (0..500),
        -- units = kilometerperhour, range (0..500), resolution = 1
        windSpeedKnots              [1] INTEGER (0..200)
        -- units = knots, range (0..200), resolution = 1
        }

SurfaceWindVariations ::= SEQUENCE
        {
        direction1                  [0] SurfaceWindDirection        OPTIONAL,
        direction2                  [1] SurfaceWindDirection        OPTIONAL,
        speedMax                    [2] SurfaceWindSpeed            OPTIONAL
        }
Temperature ::= CHOICE
        {
        notAvailable                [0] NULL,
        temperature                 [1] INTEGER (-80..60)
        -- units = degree Celsius, range (-80..60), resolution = 1
        }

Time ::= SEQUENCE
        {
        timeHours                   TimeHours,
        timeMinutes                 TimeMinutes
        }

TimeHours ::= INTEGER (0..23)
-- units = hours, range (0..23), resolution = 1

TimeMinutes ::= INTEGER (0..59)
-- units = minutes, range (0..59), resolution = 1

TimeSeconds ::= INTEGER (0..59)
-- units = seconds, range (0..59), resolution = 1
```

```
TransitionLevel ::= CHOICE
        {
        flightLevelMeter            [0] INTEGER (100..2000),
        -- units = meters, range (1000..20000), resolution = 10 meters
        flightLevelFeet             [1] INTEGER (30..600)
        -- units = feet, range (3000..60000), resolution = 100 feet
        }

VerticalVisibility ::= CHOICE
        {
        visibilityMeters            [0] INTEGER (0..100),
        -- units = meters, range (0..3000), resolution = 30
        visibilityFeet              [1] INTEGER (0..100),
        -- units = feet, range (0..10000), resolution = 100
        notAvailable                [2] NULL
        }

VisibilityStatuteMiles ::= CHOICE
        {
        oneSixteenth                [0] INTEGER (0..6),
        -- units = Statute Mile, range (0..3/8), resolution = 1/16th
        oneEighth                   [1]  INTEGER (3..16),
        -- units = Statute Mile, range (3/8..2), resolution = 1/8th
        oneFourth                   [2]  INTEGER (8..12),
        -- units = Statute Mile, range (2..3), resolution = 1/4th
        one                         [3]  INTEGER (3..15),
        -- units = Statute Mile, range (3..15), resolution = 1
        five                        [4]  INTEGER (3..10),
        -- units = Statute Mile, range (15..50), resolution = 5
        moreThanFifty               [5] NULL
        }

Visibility ::= CHOICE
        {
        lowMeter                    [0] INTEGER (0..10),
        -- units = meters, range (0..500), resolution = 50
        highMeter                   [1] INTEGER (6..49),
        -- units = meters, range (600..4900), resolution = 100
        kms                         [2] INTEGER (5..10),
        -- units = kms, range (5..10), resolution = 1
        statuteMiles                [3] VisibilityStatuteMiles
        }

Year ::= INTEGER (1996..2095)
-- unit = year, range (1996..2095), resolution = 1
```

END

2.4.5 PROTOCOL DEFINITION

2.4.5.1 Sequence Rules

2.4.5.1.1 With the exception of abort primitives, only the sequence of primitives illustrated in figures 2.4.5-1 to 2.4.5-17 shall be permitted.

Note 1.— The following figures define the valid sequences of primitives that is possible to invoke during the operation of the FIS application. 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.— All abort primitives may interrupt and terminate any of the normal message sequences outline below.

Note 3.— Primitives are processed in the order in which they are received (see 4.3.3).

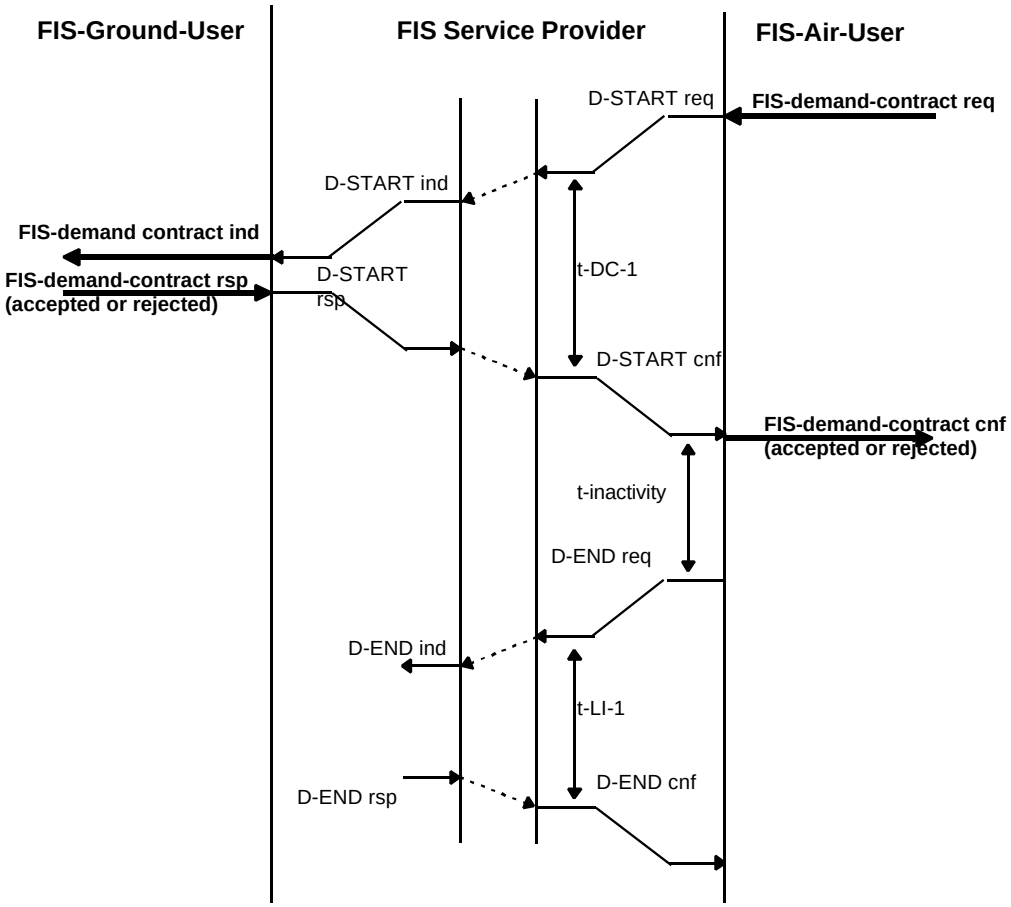

Figure 2.4.5-1. Use of FIS-demand-contract service with no dialogue existing with contract accept or contract reject

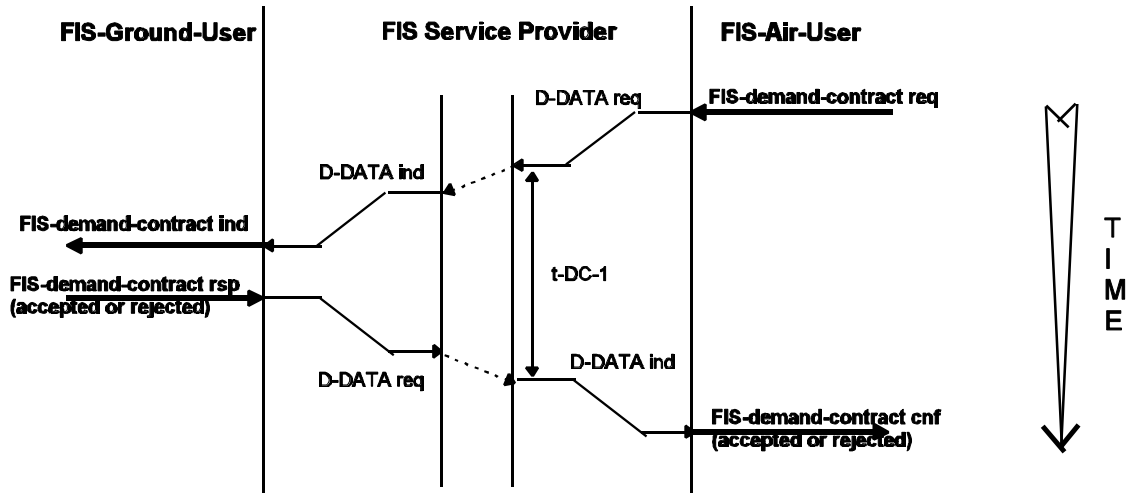

Figure 2.4.5-2. Use of FIS-demand-contract service with dialogue existing with contract accept or reject

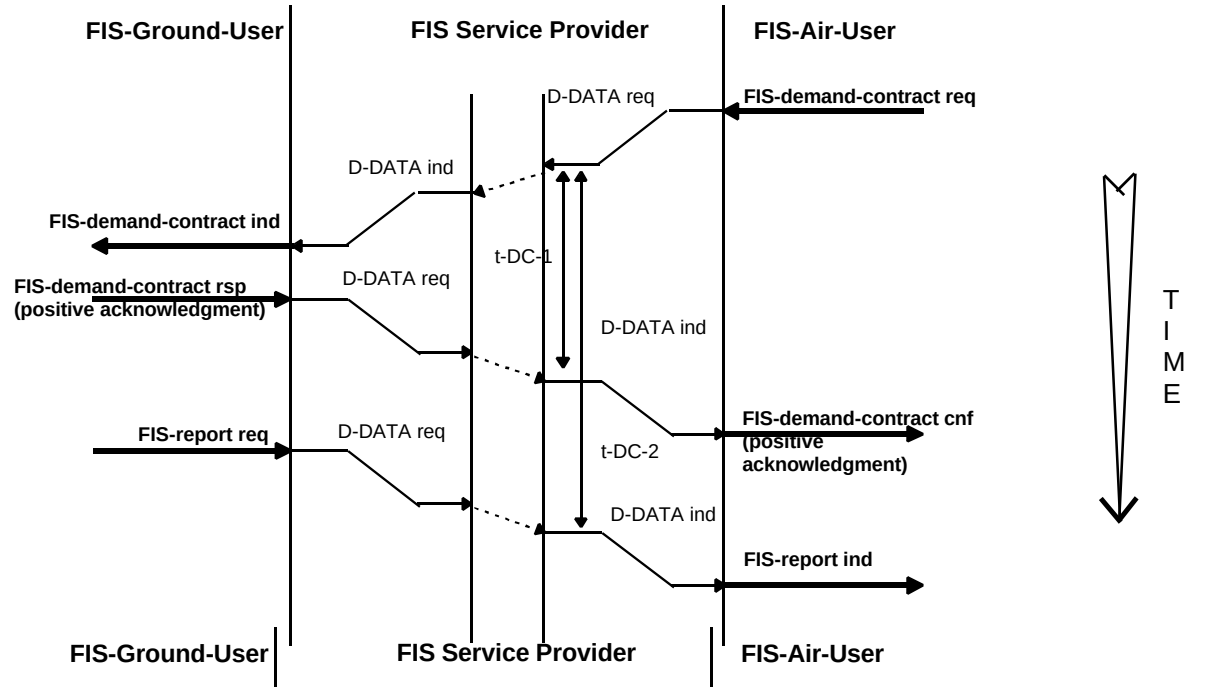

Figure 2.4.5-4. Use of FIS-demand-contract with existing dialogue and positive acknowledgement

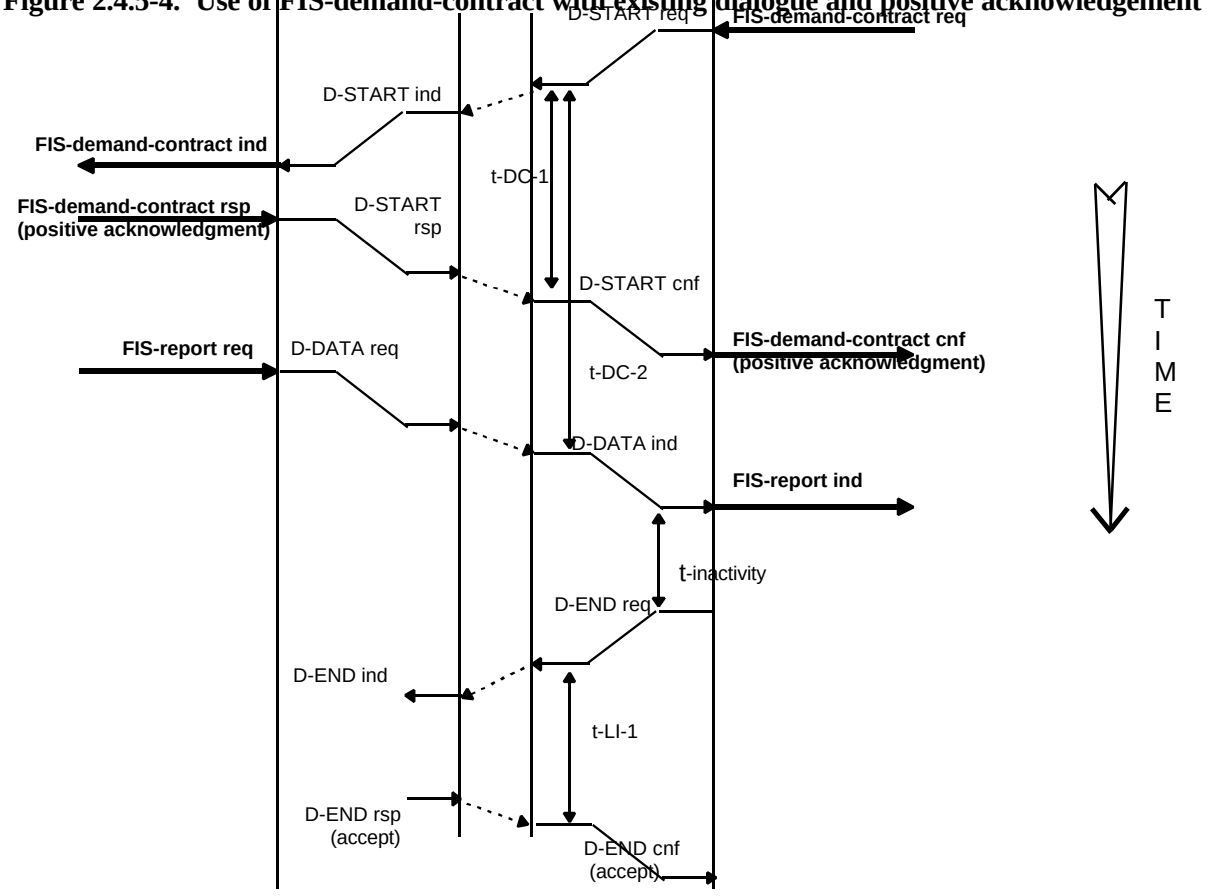

Figure 2.4.5-3. Use of FIS-demand-contract service with no existing dialogue and with a positive acknowledgement

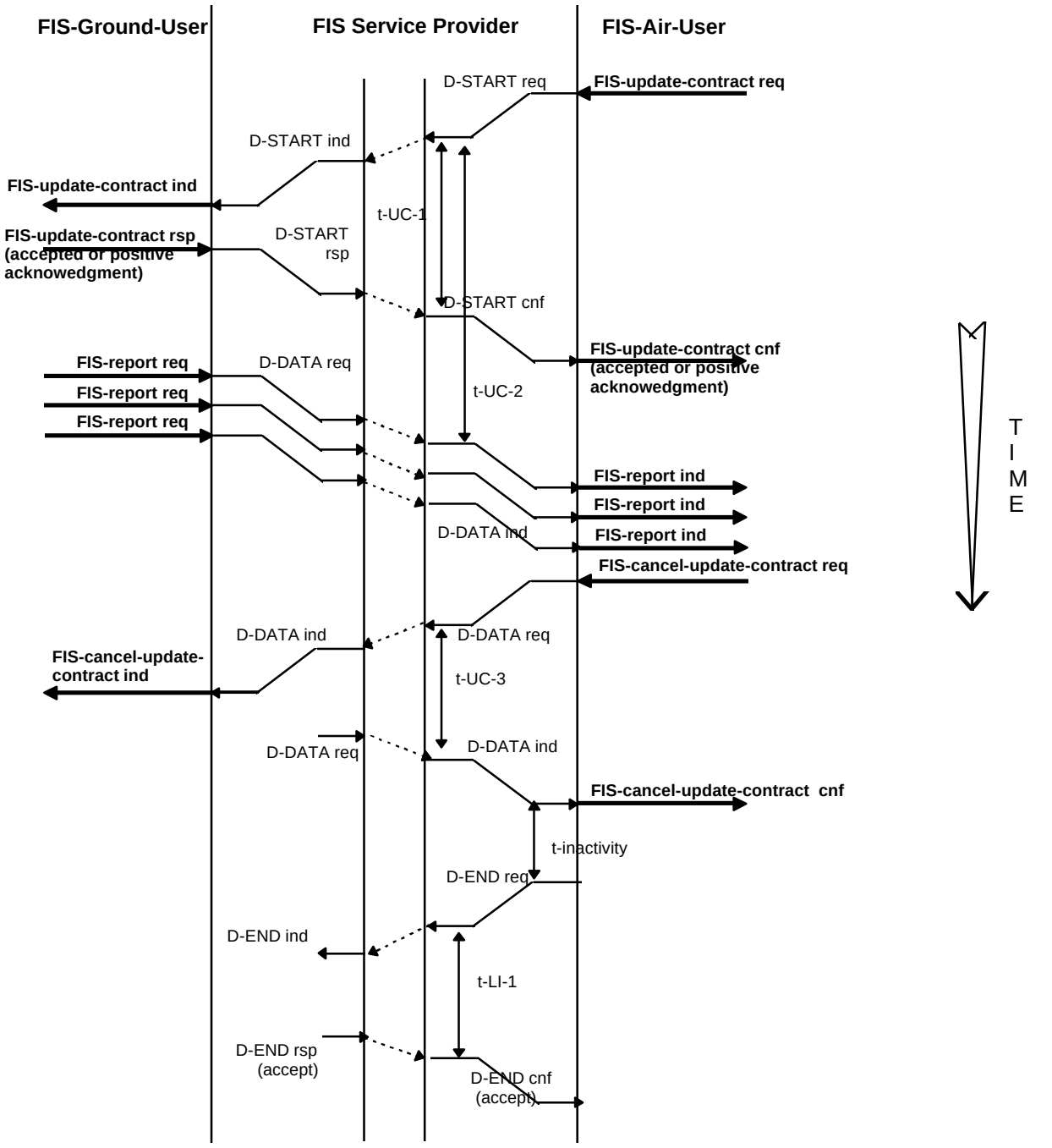

Figure 2.4.5-5. Use of FIS-update-contract service with no existing dialogue and with contract accept and with air-initiated cancellation

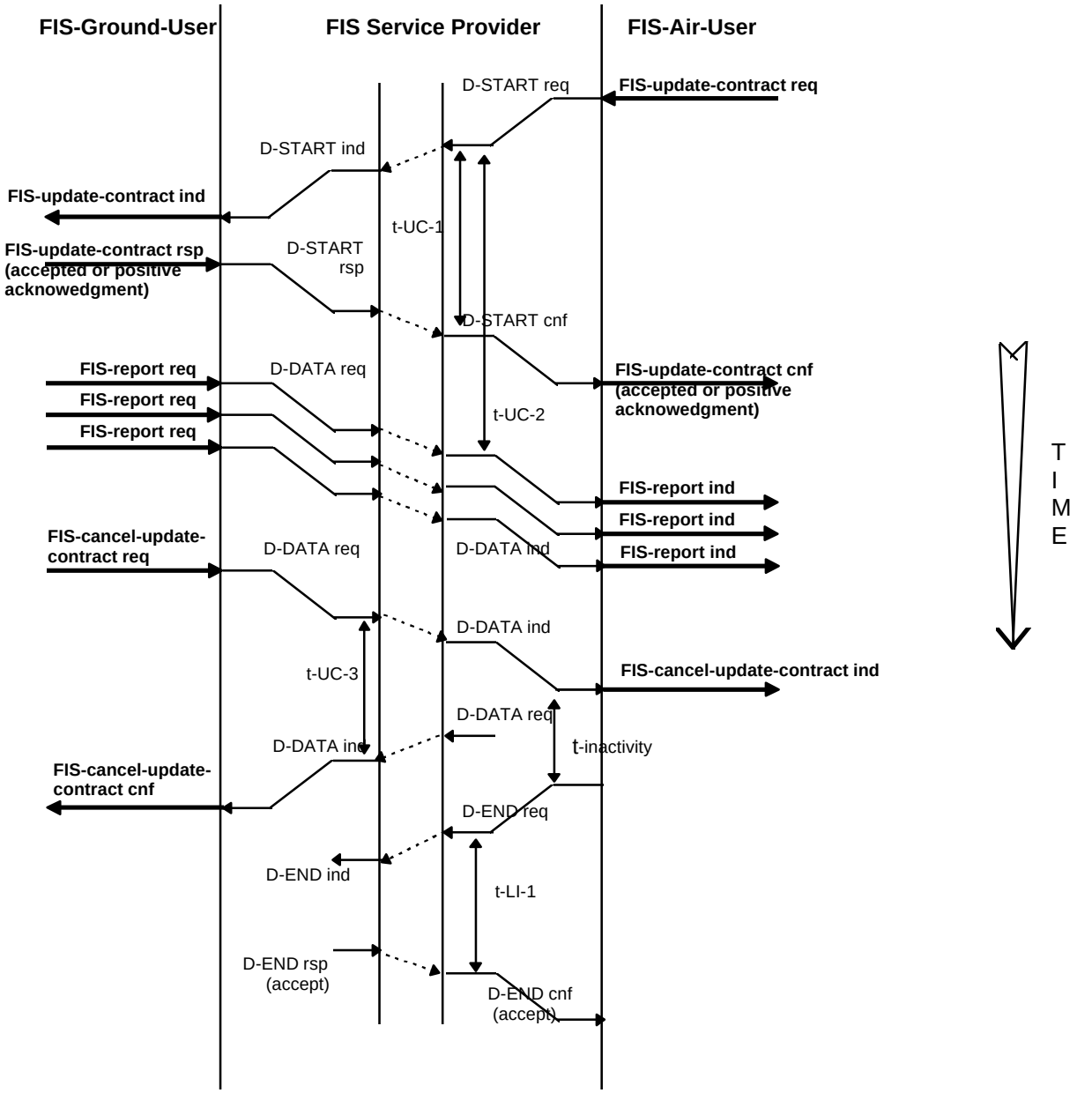

Figure 2.4.5-6. Use of FIS-update-contract service with no existing dialogue and with contract accept and with ground-initiated cancellation

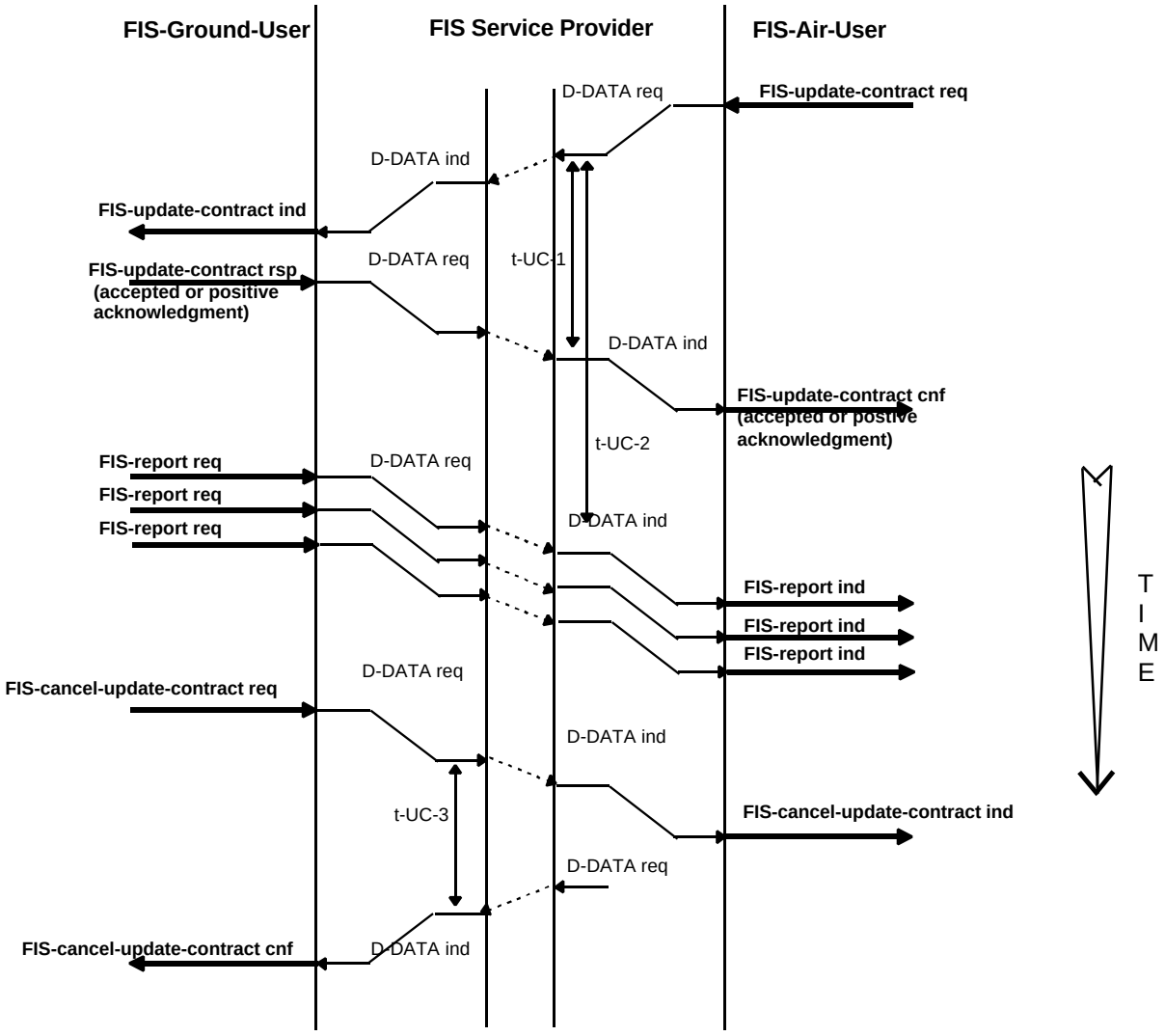

Figure 2.4.5-7. Use of FIS-update-contract service with existing dialogue and with contract accept and with ground-initiated cancellation

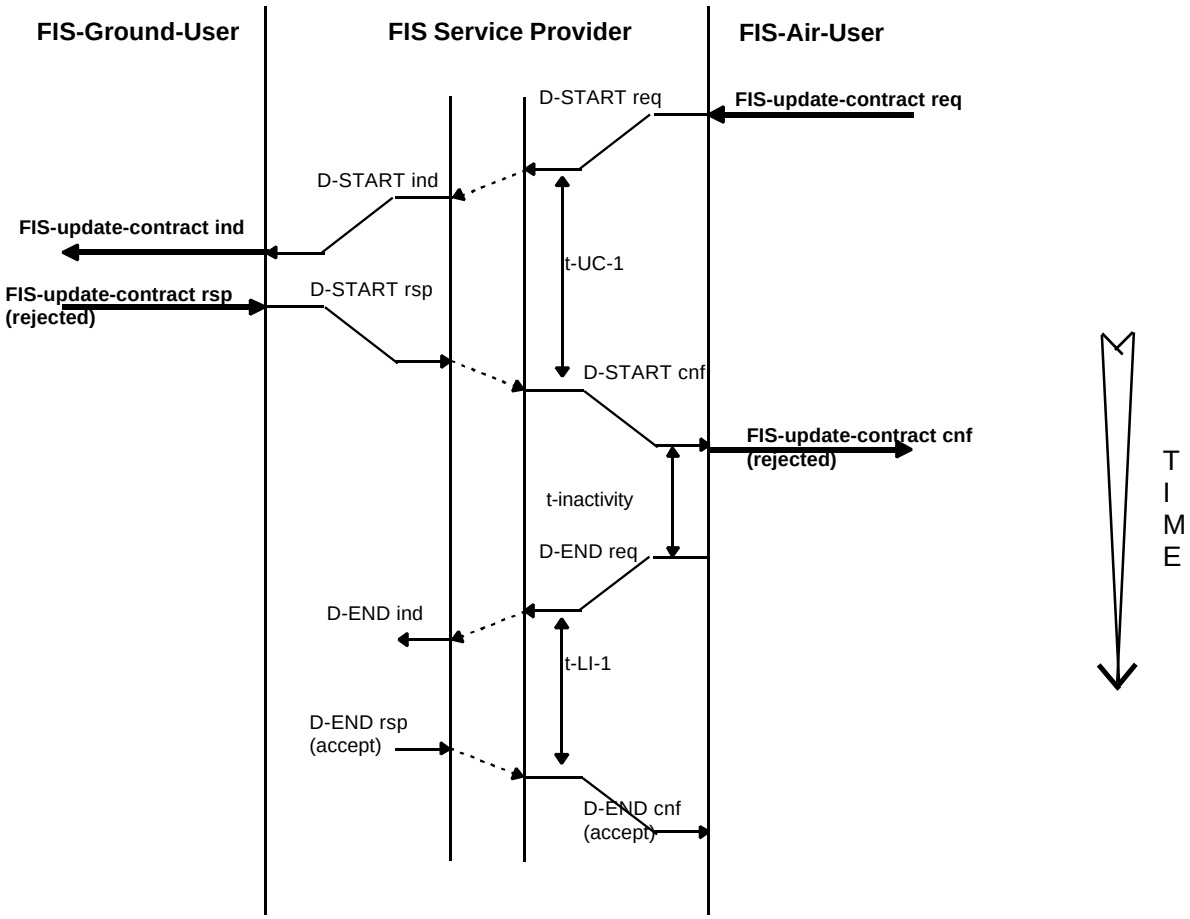

Figure 2.4.5-8. Use of the FIS-update-contract service with no existing dialogue and with contract reject

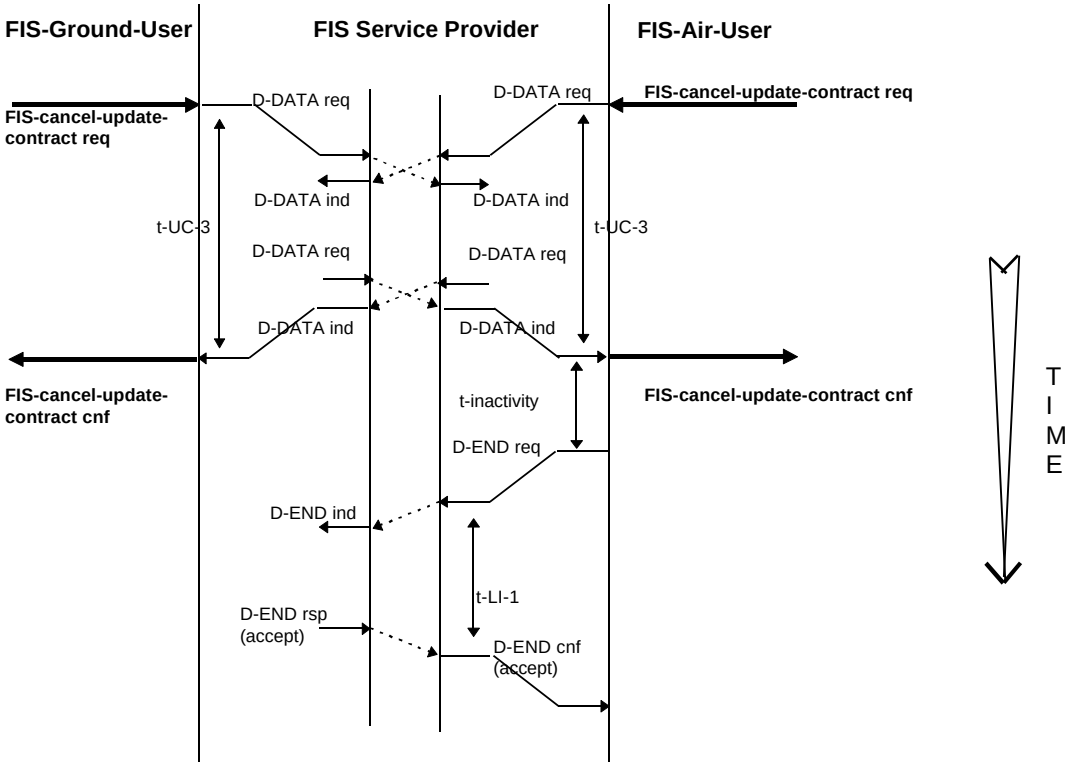

Figure 2.4.5-10. Use of FIS-cancel-update service by both FIS-users without other FIS Contract in place

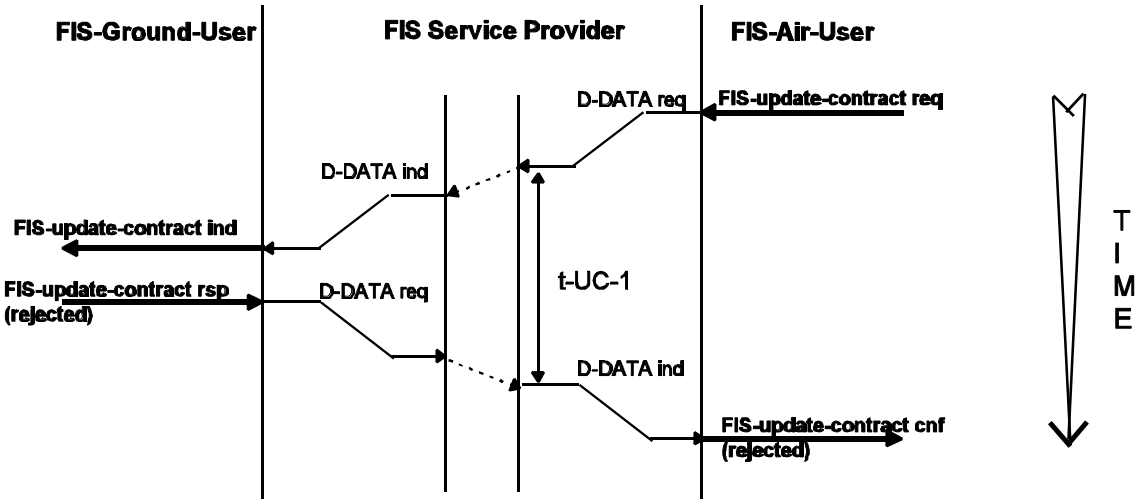

Figure 2.4.5-9. Use of FIS-update-contract service with existing dialogue and with contract reject

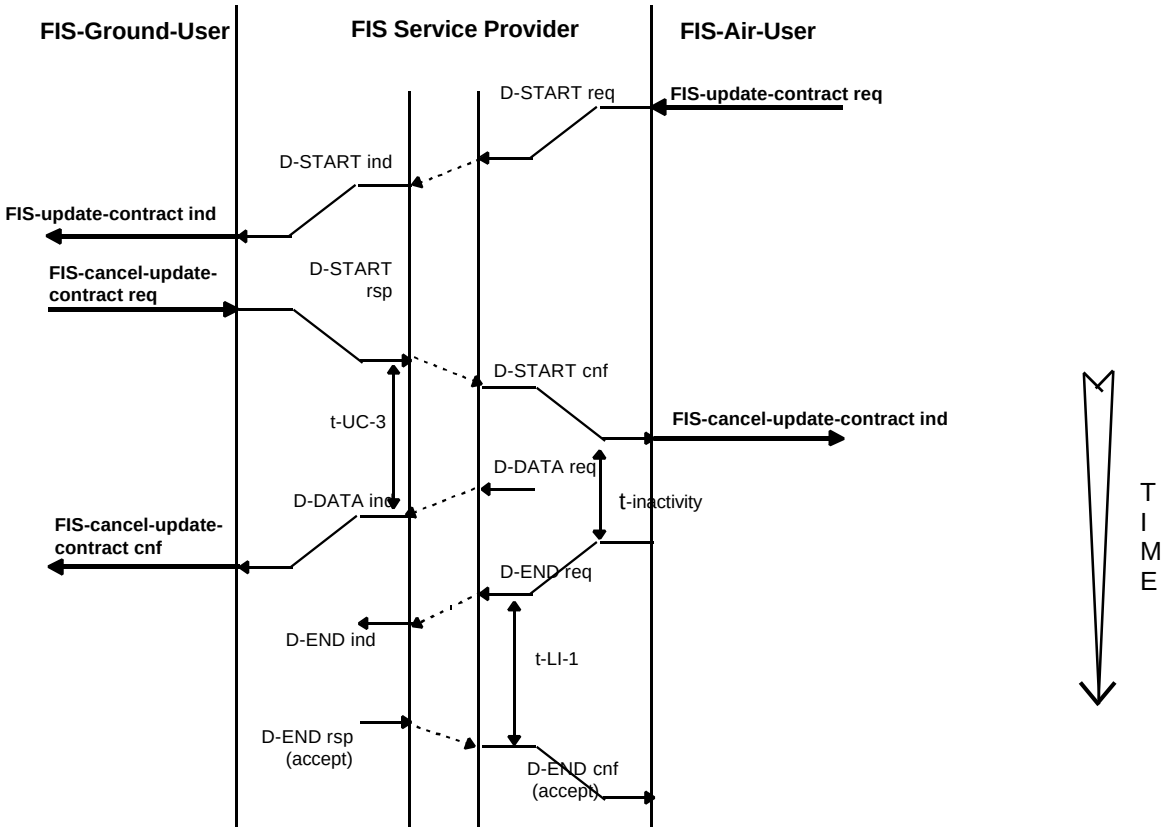

Figure 2.4.5-11. Use of FIS-update-contract service with no existing dialogue and with ground-initiated cancellation during contract establishment phase

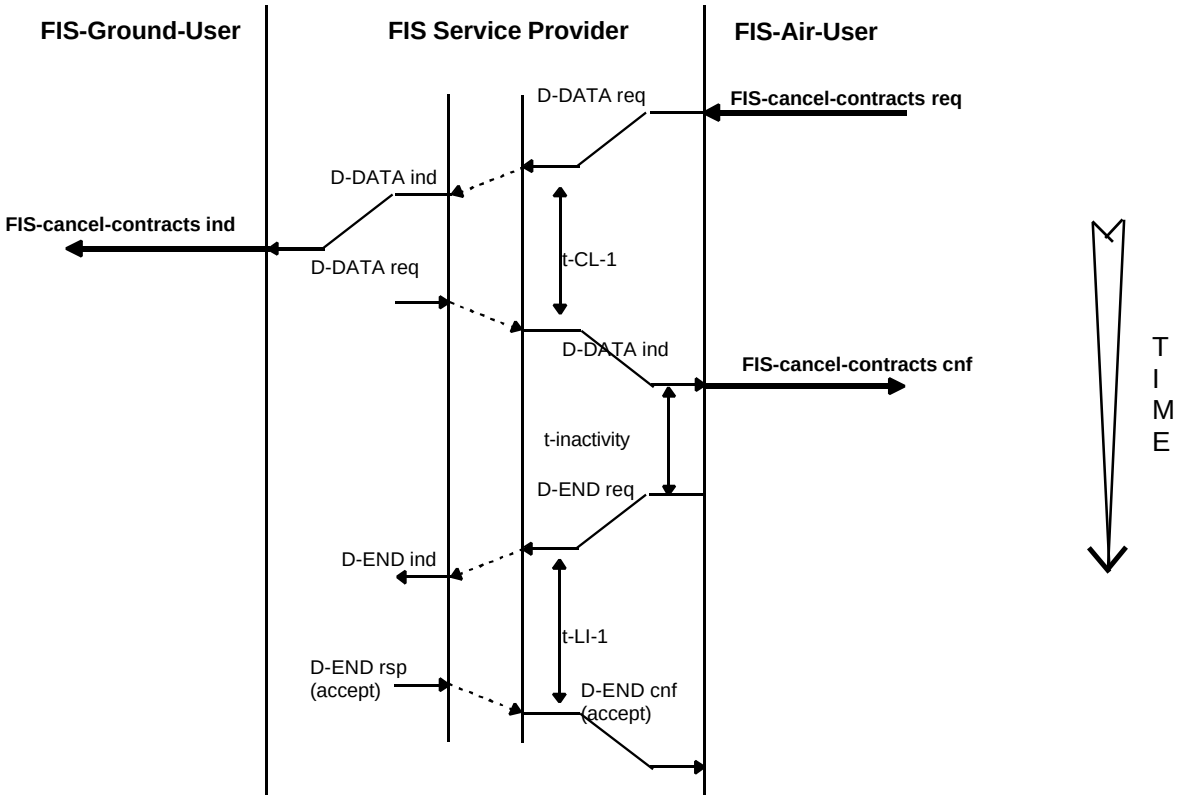

Figure 2.4.5-12. Use of FIS-cancel-contracts service with all contracts in place cancelled

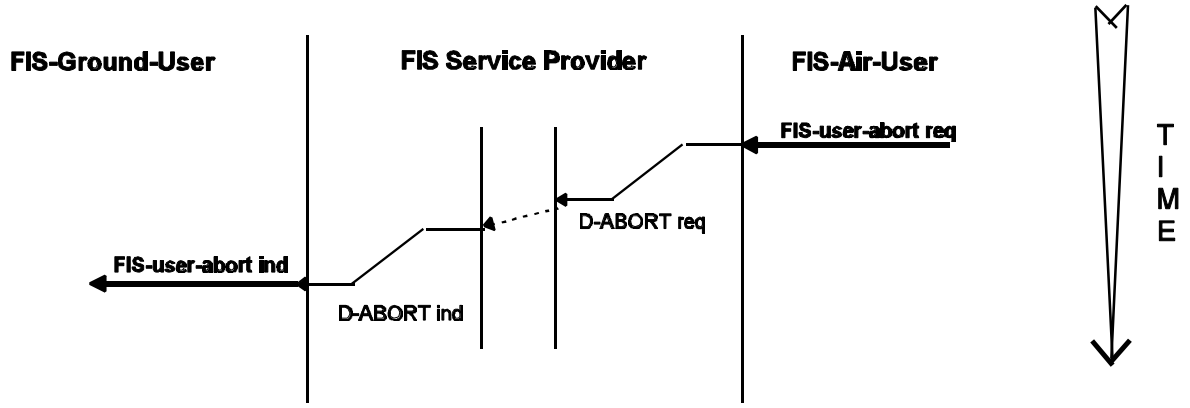

Figure 2.4.5-13. Use of FIS-user-abort service (air-initiated)

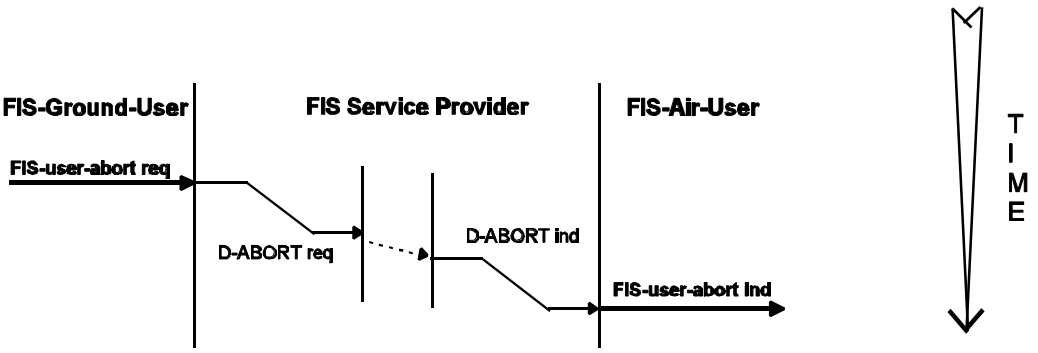

Figure 2.4.5-14. Use of FIS-user-abort service (ground-initiated)

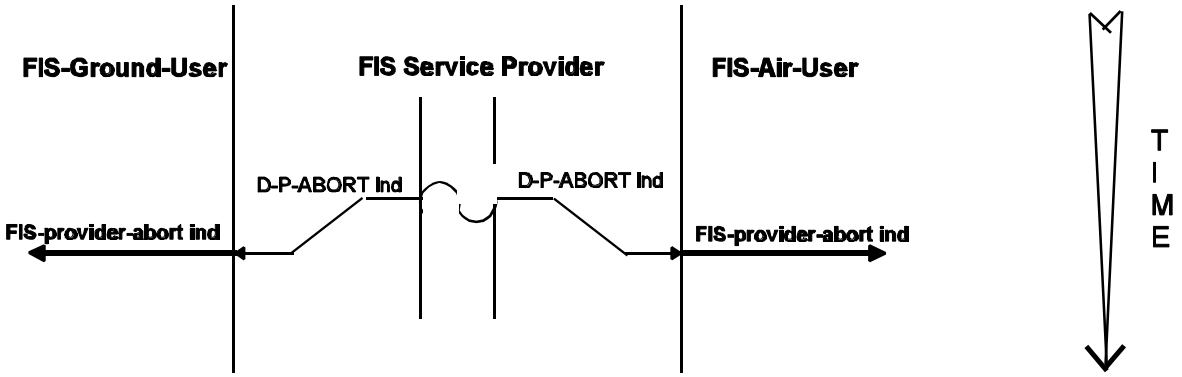

Figure 2.4.5-15. Use of FIS-provider-abort service (dialogue service provider aborts)

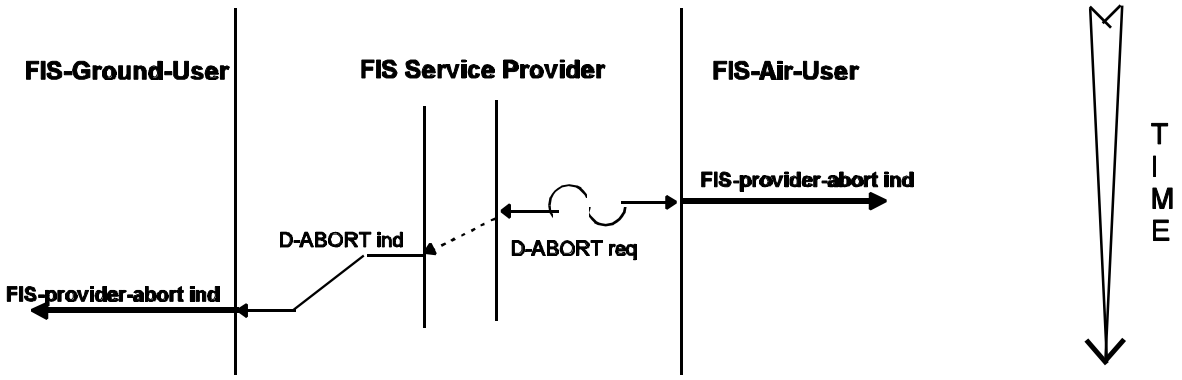

Figure 2.4.5-16. Use of FIS-provider-abort service (FIS-air-ASE aborts)

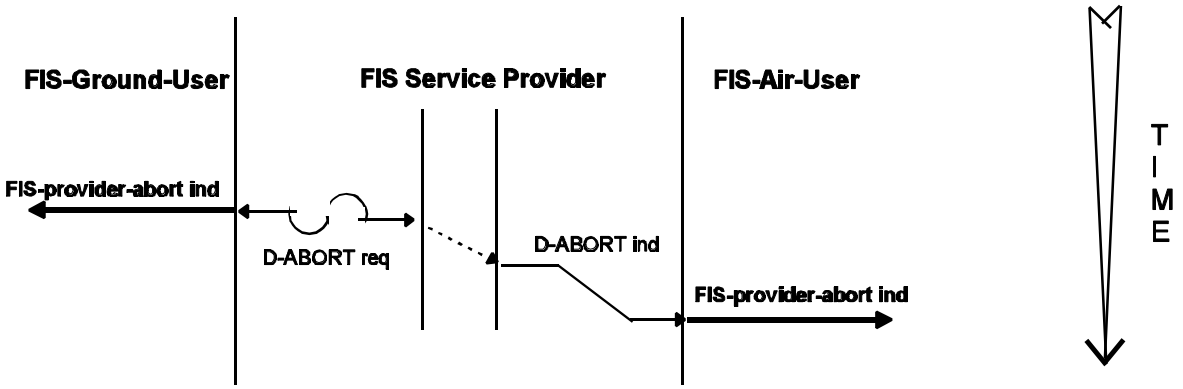

Figure 2.4.5-17. Use of FIS-provider-abort service (FIS-ground-ASE aborts)

2.4.5.2 FIS Service Provider Timers

2.4.5.2.1 The FIS-ASEs shall be capable of detecting when a timer expires.

Note 1.— Table 2.4.5-1 lists the time constraints related to the FIS application. Each time constraint requires a timer to be set in the FIS protocol machine.

Note 2.— If the timer expires before the final event has occurred:

a) for all timers but the t-inactivity timer, the FIS-ASEs takes the appropriate action as defined in 2.4.5.3.

b) for the t-inactivity timer, the FIS-air-ASE closes the dialogue by using the D-END service.

2.4.5.2.2 **Recommendation. —** *The timer values should be as indicated in Table 2.4.5-1.*

Table 2.4.5-1. FIS Service-Provider Timers

FIS Service	Timer Name	Timer Value	Timer Start Event	Timer Stop Event
FIS-demand-contract	t-DC-1	6 min	FIS-demand-contract req	FIS-demand-contract cnf
	t-DC-2	9 min	FIS-demand-contract req	FIS-report ind
FIS-update-contract	t-UC-1	9 min	FIS-update-contract req	FIS-update-contract cnf
	t-UC-2	6 min	FIS-update-contract req	FIS-report ind
FIS-cancel-update-contract	t-UC- 3	9 min	FIS-cancel-update-contract req	FIS-cancel-update-contract cnf

FIS Service	Timer Name	Timer Value	Timer Start Event	Timer Stop Event
FIS-cancel-contracts	t-CL-1	6 min	FIS-cancel-contracts req	FIS-cancel-contracts cnf
General	t-LI-1	6 min	D-END req	D-END cnf
	t-inactivity	(see note)	last primitive of the last contract received send by the FIS-air-ASE to the FIS-air-user	FIS-demand-contract req or FIS-update-contract req

Note 1.— The t-inactivity timer value is set on configuration basis.

Note 2.— The receipt of FIS-user-abort request, D-ABORT indication and D-P-ABORT indication are also timer stop events.

2.4.5.3 FIS-ASE Protocol Description

2.4.5.3.1 Functional Model

Note 1.— The FIS-ASE is functionally made of 6 modules, as shown in Figure 2.4.5-18:

*a) the **FIS High Interface module** (HI module).This module interfaces with the ASE-user through the abstract service interface as defined in 2.4.3.*

*b) the **FIS Demand Contract module** (DC module). This module manages a single FIS Demand Contract.*

*c) the **FIS Update Contract module** (UC module). This module manages a single FIS Update Contract.*

*d) the **FIS Cancel contracts module** (CL module). This module processes the termination of all contracts of the same type (i.e. ATIS) still in operation.*

*e) the **FIS Abort module** (AB module). This module handles aborts in case of unrecoverable error.*

*f) the FIS **Low Interface module** (LI module). This module interfaces the Dialogue Service Provider on behalf of the DC, UC, CL and AB modules. It performs the multiplexing of FIS Contracts on a single dialogue.*

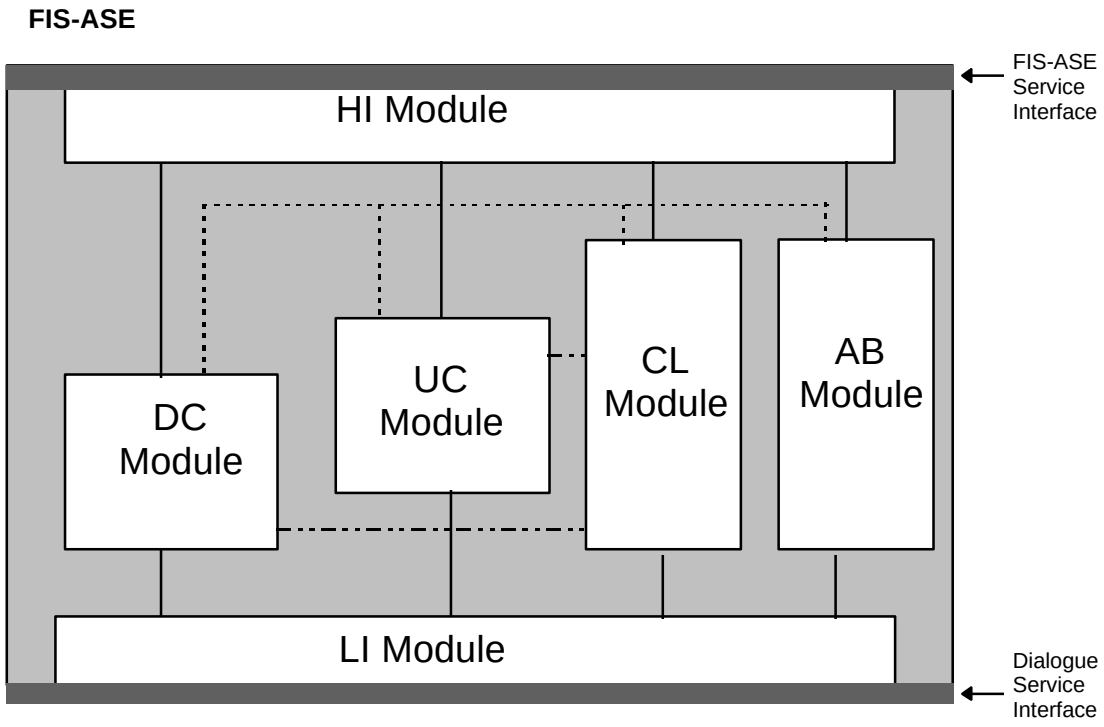

Figure 2.4.5-18. Functional Architecture of the FIS-air-ASE and the FIS-ground-ASE

Note 2.— This functional architecture allows simplification of the description of the protocol handled by the FIS-ASE. It does not constrain the implementation architecture.

Note 3.— The following subsections describe the actions of the individual modules in both the air and ground ASEs. 2.4.5.5 contains state tables for the individual modules.

Note 4.— The FIS-air-user is considered an active user from the time at which it invokes the first FIS-demand-contract request or FIS-update-contract request until such time that:

a) the FIS-air-ASE invokes a FIS-demand-contract confirmation (accepted or rejected) and there are no contracts in place,

b) the FIS-air-ASE invokes a FIS-report indication following a FIS-demand-contract confirmation (positive acknowledgment),

c) the FIS-air ASE invokes a FIS-update-contract confirmation (rejected) and there is no contract in place,

d) the FIS-air-ASE invokes a FIS-cancel-update-contract confirmation and there is no contract in place,

e) the FIS-air-ASE invokes a FIS-cancel-update-contract indication and there is no contract in place,

f) the FIS-air-ASE invokes a FIS-cancel-contracts confirmation,

g) the FIS-air-ASE receives a FIS-user-abort request,

h) the FIS-air-user invokes a FIS-user-abort indication, or

i) the FIS-air-user receives a FIS-user-abort indication.

Note 5.— The FIS-ground-user is considered an active user from the time at which it receives the first FIS-demand-contract indication or FIS-update-contract indication until such time that:

a) the FIS-ground-ASE receives a FIS-demand-contract response (accepted or rejected) and there is no contract in place,

b) the FIS-ground-ASE receives a FIS-report request following a FIS-demand-contract response (positive acknowledgment),

c) the FIS-ground-ASE receives a FIS-update-contract response (rejected) and there is no contract in place,

d) the FIS-ground-ASE invokes a FIS-cancel-update-contract indication and there is no contract in place,

e) the FIS-ground-ASE invokes a FIS-cancel-update-contract confirmation and there is no contract in place,

f) the FIS-ground-ASE invokes a FIS-cancel-contracts indication,

g) the FIS-ground-ASE receives a FIS-user-abort request,

h) the FIS-ground-user invokes a FIS-user-abort indication, or

i) the FIS-ground-user receives a FIS-user-abort indication.

2.4.5.3.2 In the following sections, if no actions are described for a FIS-service primitive in a particular state, then the invocation of that service primitive shall be prohibited in that state.

2.4.5.3.3 Possible Errors Arising from Receipt of an APDU

2.4.5.3.3.1 Upon receipt of an APDU, if no actions are described for the arrival of that APDU when in a particular state, then exception handling procedures as described in 2.4.5.4.3 shall apply.

2.4.5.3.3.2 If an APDU is not received when one is required, then exception handling procedures as described in 2.4.5.4.2 shall apply.

2.4.5.3.3.3 Upon receipt of an APDU that cannot be decoded, exception handling procedures as described in 2.4.5.4.4 shall apply.

2.4.5.3.4 Ground and Air FIS HI Module

Note.— All statements in 2.4.5.3.4 apply to both the FIS ground HI module and the FIS air HI module.

2.4.5.3.4.1 Upon receipt of a FIS-ASE service request or response primitive, the FIS HI module shall:

a) if the primitive is a FIS-demand-contract request or a FIS-update-contract request primitive,

1) reject the primitive if the *FISContractNumber* parameter corresponds to an existing FIS contract,

2) else stop timer t-inactivity if set,

b) if the primitive is not a FIS-demand-contract request or a FIS-update-contract request primitive, reject the primitive if the *FISContractNumber* parameter does not correspond to an existing FIS contract,

c) otherwise pass the primitive to the relevant DC, UC, CL or AB module, as shown in Tables 2.4.5-2/a and 2.4.5-2/b.

Table 2.4.5-2/a. Request and response primitive to FIS-ASE module mapping - Air ASE

FIS-ASE Service Primitive Name	**FIS-ASE Module**
FIS-demand-contract req	DC
FIS-update-contract req	UC
FIS-cancel-update-contract req	UC
FIS-user-abort req	AB
FIS-cancel-contracts req	CL

Table 2.4.5-2/b. Request and response primitive to FIS-ASE module mapping - Ground ASE

FIS-ASE Service Primitive Name	FIS-ASE Module
FIS-demand-contract rsp	DC
FIS-update-contract rsp	UC
FIS-cancel-update-contract req	UC
FIS-user-abort req	AB
FIS-report req	UC or DC based on the FISContractNumber

2.4.5.3.4.2 Upon receipt of a request to invoke a service indication or confirmation primitive from one of the modules in the FIS-ASE as shown in Tables 2.4.5-3/a and 2.4.5-3/b, and if the FIS-user is active, the FIS HI module shall do so.

Table 2.4.5-3/a: FIS-ASE module to indication and confirmation primitives mapping - Air ASE

FIS-ASE Module	FIS-ASE Service Primitive Name
DC	FIS-demand-contract cnf
UC	FIS-update-contract cnf
UC	FIS-cancel-update-contract ind
UC	FIS-cancel-update-contract cnf
DC, UC	FIS-report ind
AB	FIS-user-abort ind
AB	FIS-provider-abort ind
CL	FIS-cancel-contracts cnf

Table 2.4.5-3/b: FIS-ASE module to indication and confirmation primitives mapping - Ground ASE

FIS-ASE Module	FIS-ASE Service Primitive Name
DC	FIS-demand-contract ind
UC	FIS-update-contract ind
UC	FIS-cancel-update-contract ind
UC	FIS-cancel-update-contract cnf
AB	FIS-user-abort ind
AB	FIS-provider-abort ind
CL	FIS-cancel-contracts ind

2.4.5.3.4.3 The air HI module shall reject requests and responses, apart from FIS-user-abort requests, when the air LI module is in the LI-START-I state or the LI-END-I state.

2.4.5.3.5 Air FIS DC Module

Note.— The states defined for the air FIS DC module are the following:

a) DC-A-IDLE,

b) DC-A-PENDING, and

c) DC-A-ACTIVE.

2.4.5.3.5.1 On initiation, the air FIS DC module shall be in the DC-A-IDLE state.

2.4.5.3.5.2 Upon receipt of a FIS-demand-contract request, then

2.4.5.3.5.2.1 If in the DC-A-IDLE state, the air FIS DC module shall:

a) start timers t-DC-1 and t-DC-2,

b) create a FISDownlinkAPDU [FISRequest] APDU with a *demandContract* APDU-element based on the value of the *FISContractNumber* and *FISContractDetails* parameters,

c) request the LI module to send the APDU to the FIS-ground-ASE identified by the value of the received *ICAOFacilityDesignation* parameter, and with the *ClassOfCommunicationService* parameter value if specified by the user, and

d) enter the DC-A-PENDING state.

2.4.5.3.5.3 Upon receipt of a [FISAccept] APDU with an *accept* APDU-element, then

2.4.5.3.5.3.1 If in the DC-A-PENDING state, the air FIS DC module shall:

a) stop timers t-DC-1 and t-DC-2,

b) if there is no other FIS contract in place, start t-inactivity timer,

c) request the FIS HI module to invoke a FIS-demand-contract confirmation with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element,

2) the *Result* parameter, containing the abstract value "accepted", and

3) the *FISInformation* parameter containing the information which has been received as the *FISReportData* APDU-element, and

d) enter the DC-A-IDLE state.

2.4.5.3.5.4 Upon receipt of a [FISAccept] APDU containing a *positiveAcknowledgement* APDU-element, then

2.4.5.3.5.4.1 If in the DC-A-PENDING state, the air FIS DC module shall:

a) stop timer t-DC-1,

b) request the FIS HI module to invoke a FIS-demand-contract confirmation with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

2) the *Result* parameter, containing the abstract value "positive acknowledgement", and

c) enter the DC-A-ACTIVE state.

2.4.5.3.5.5 Upon receipt of a [FISReject] APDU, then

2.4.5.3.5.5.1 If in the DC-A-PENDING state, the air FIS DC module shall:

a) stop timers t-DC-1 and t-DC-2 ,

b) if there is no other FIS contract in place, start t-inactivity timer,

c) request the FIS HI module to invoke a FIS-demand-contract confirmation with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the ContractNumber APDU-element,

2) the *Result* parameter, containing the abstract value “rejected”, and

3) the *RejectReason* parameter containing the information which has been received as the *FISRejectReason* APDU-element, and

d) enter the DC-A-IDLE state.

2.4.5.3.5.6 Upon receipt of a [FISReport] APDU, then

2.4.5.3.5.6.1 If in the DC-A-ACTIVE state, the air FIS DC module shall:

a) stop timer t-DC-2 ,

b) if there is no other FIS contract in place, start t-inactivity timer,

c) request the FIS HI module to invoke a FIS-report indication with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

2) the *FISInformation* parameter, containing the information which has been received as the *FISReportData* APDU-element, and

d) enter the DC-A-IDLE state.

2.4.5.3.5.7 Upon receipt of a request from the AB module to stop operation, then

2.4.5.3.5.7.1 The air FIS DC module shall:

a) stop any timers, and

b) enter the DC-A-IDLE state.

2.4.5.3.5.8 Upon expiration of the t-DC-1 timer, then

2.4.5.3.5.8.1 If in the DC-A-PENDING state, the air FIS DC module shall:

a) stop timer t-DC-2,

b) request the AB module to abort with reason “timer expiration”, and

c) enter the DC-A-IDLE state.

2.4.5.3.5.9 Upon expiration of the t-DC-2 timer, then

2.4.5.3.5.9.1 If in the DC-A-ACTIVE state, the air FIS DC module shall:

a) request the AB module to abort with reason “timer expiration”, and

b) enter the DC-A-IDLE state.

2.4.5.3.6 Ground FIS DC Module

Note.— The states defined for the ground FIS DC module are the following:

a) DC-G-IDLE,

b) DC-G-PENDING, and

c) DC-G-ACTIVE.

2.4.5.3.6.1 On initiation, the ground FIS DC module shall be in the DC-G-IDLE state.

2.4.5.3.6.2 Upon receipt of a [FISRequest] APDU with a *demandContract* APDU-element, then

2.4.5.3.6.2.1 If in the DC-G-IDLE state, the ground FIS DC module shall:

a) request the FIS HI module to invoke a FIS-demand-contract indication with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

2) the *FISContractDetails* parameter, containing the information which has been received as the *FISRequestData* APDU-element, and

b) enter the DC-G-PENDING state.

2.4.5.3.6.3 Upon receipt of a FIS-demand-contract response with the *Result* parameter containing the abstract value “accepted”, then

2.4.5.3.6.3.1 If in the DC-G-PENDING state, the ground FIS DC module shall:

a) create a FISUplinkAPDU [FISAccept] APDU with an *accept* APDU-element based on the value of the *FISContractNumber* and *FISInformation* parameters,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the DC-G-IDLE state.

2.4.5.3.6.4 Upon receipt of a FIS-demand-contract response with the *Result* parameter containing the abstract value "positive acknowledgement", the ground FIS DC module shall:

a) create a FISUplinkAPDU [FISAccept] APDU with a *positiveAcknowledgement* APDU-element based on the value of the *FISContractNumber* parameter,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the DC-G-ACTIVE state.

2.4.5.3.6.5 Upon receipt of a FIS-demand-contract response with the *Result* parameter containing the abstract value "rejected", the ground FIS DC shall:

a) create a FISUplinkAPDU [FISReject] APDU with an *otherReasons* APDU-element based on the value of the *FISContractNumber* and the *RejectReason* parameters,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the DC-G-IDLE state.

2.4.5.3.6.6 Upon receipt of a FIS-report request, then

2.4.5.3.6.6.1 If in the DC-G-ACTIVE state, the ground FIS DC module shall:

a) create a FISUplinkAPDU [FISReport] APDU based on the value of the *FISContractNumber* and the *FISInformation* parameters,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the DC-G-IDLE state.

2.4.5.3.6.7 Upon receipt of a request from the AB module to stop operation, then

2.4.5.3.6.7.1 The ground FIS DC module shall enter the DC-G-IDLE state.

2.4.5.3.7 Air FIS UC Module

Note.— The states defined for the air UC module are the following:

a) *UC-A-IDLE,*

b) *UC-A-PENDING,*

c) *UC-A-ACTIVE, and*

d) *UC-A-CANCEL*

2.4.5.3.7.1 On initiation, the air FIS UC module shall be in the UC-A-IDLE state.

Note.— The air FIS UC module has a boolean variable named CANCELFROMPENDING.

2.4.5.3.7.2 On initiation, CANCELFROMPENDING shall be set to FALSE.

2.4.5.3.7.3 Upon receipt of a FIS-update-contract request, then

2.4.5.3.7.3.1 If in the UC-A-IDLE state, the air FIS UC module shall:

a) start timers t-UC-1 and t-UC-2,

b) create a FISDownlinkAPDU [FISRequest] APDU with an *updateContract* APDU-element based on the value of the *FISContractNumber* and *FISContractDetails* parameters,

c) request the LI module to send the APDU to the FIS-ground-ASE identified by the value of the received *ICAOFacilityDesignation* parameter, with the *ClassOfCommunicationService* parameter value if specified by the user, and

d) enter the UC-A-PENDING state.

2.4.5.3.7.4 Upon receipt of a [FISAccept] APDU containing an *accept* APDU-element, then

2.4.5.3.7.4.1 If in the UC-A-PENDING state, the air FIS UC module shall:

a) stop timers t-UC-1 and t-UC-2,

b) request the FIS HI module to invoke a FIS-update-contract confirmation with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element,

2) the *Result* parameter, containing the abstract value “accepted”, and

3) the *FIS Information* parameter, containing the information which has been received as the *FISReportData* APDU-element, and

c) enter the UC-A-ACTIVE state.

2.4.5.3.7.4.2 If in the UC-A-CANCEL state and the CANCELFROMPENDING is set to TRUE, the air FIS UC module shall:

a) stop timers t-UC-1 and t-UC-2,

b) set CANCELFROMPENDING to FALSE, and

c) remain in the UC-A-CANCEL state.

2.4.5.3.7.5 Upon receipt of a [FISAccept] APDU containing a *positiveAcknowledgement* APDU-element, then

2.4.5.3.7.5.1 If in the UC-A-PENDING state, the air FIS UC module shall:

a) stop timer t-UC-1,

b) request the FIS HI module to invoke a FIS-update-contract confirmation with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

2) the *Result* parameter, containing the abstract value "positive acknowledgement", and

c) enter the UC-A-ACTIVE state.

2.4.5.3.7.5.2 If in the UC-A-CANCEL state and the CANCELFROMPENDING is set to TRUE, the air FIS UC module shall:

a) stop timers t-UC-1 and t-UC-2,

b) set CANCELFROMPENDING to FALSE, and

c) remain in the UC-A-CANCEL state.

2.4.5.3.7.6 Upon receipt of a [FISReject] APDU, then:

2.4.5.3.7.6.1 If in the UC-A-PENDING state, the air FIS UC module shall:

a) stop timers t-UC-1 and t-UC-2 ,

b) if there is no other FIS contract in place, start t-inactivity timer,

c) request the FIS HI module to invoke a FIS-update-contract confirmation with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element,

2) the *Result* parameter, containing the abstract value "rejected",

3) the *RejectReason* parameter containing:

i) the information which has been received as the *FISRejectReason* APDU-element, if the FISRejectData contains the *otherReasons* element, or

ii) the abstract value "update contract function not supported by the FIS-ground-user", if the *FISRejectData* APDU-element contains the *updateFISContractNotSupported* or *updateFISContractNotSupportedWithReport* choice, and

4) the *FISInformation* parameter containing the information which has been received as the *FISReportData* element, if the *FISRejectData* APDU-element contains the *updateFISContractNotSupportedWithReport* choice, and

d) enter the UC-A-IDLE state.

2.4.5.3.7.6.2 If in the UC-A-CANCEL state and the CANCELFROMPENDING is set to TRUE, the air FIS UC module shall:

a) stop timers t-UC-1 and t-UC-2,

b) set CANCELFROMPENDING to FALSE,

c) request the FIS HI module to invoke a FIS-cancel-update-contract confirmation with the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

d) enter the UC-A-IDLE state.

2.4.5.3.7.7 Upon receipt of a [FISReport] APDU, then

2.4.5.3.7.7.1 If in the UC-A-ACTIVE state, the air FIS UC module shall:

a) if timer t-UC-2 is set, stop timer t-UC-2 ,

b) request the FIS HI module to invoke a FIS-report indication with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

2) the *FISInformation* parameter, containing the information which has been received as the *FISReportData* APDU-element, and

c) remain in the UC-A-ACTIVE state.

2.4.5.3.7.7.2 If in the UC-A-CANCEL state and the CANCELFROMPENDING is set to FALSE, the air FIS UC module shall remain in the UC-A-CANCEL state.

2.4.5.3.7.8 Upon receipt of a FIS-cancel-update-contract request, then

2.4.5.3.7.8.1 If in the UC-A-ACTIVE state, the air FIS UC module shall:

a) if timer t-UC-2 set, stop timer t-UC-2 ,

b) create a FISDownlinkAPDU [FISCancelUpdateContract] APDU based on the value of the received *FISContractNumber* parameter,

c) request the LI module to send the APDU to the FIS-ground-ASE,

d) start the t-UC-3 timer, and

e) enter the UC-A-CANCEL state.

2.4.5.3.7.8.2 If in the UC-A-PENDING state and a dialogue is fully established, the air FIS UC module shall:

a) if timer t-UC-2 set, stop timer t-UC-2 ,

b) create a FISDownlinkAPDU [FISCancelUpdateContract] APDU based on the value of the received *FISContractNumber* parameter,

c) request the LI module to send the APDU to the FIS-ground-ASE,

d) start the t-UC-3 timer,

e) set CANCELFROMPENDING to TRUE, and

f) enter the UC-A-CANCEL state.

2.4.5.3.7.9 Upon receipt of a [FISCancelUpdateContract] APDU, then

2.4.5.3.7.9.1 If in the UC-A-ACTIVE state, the air FIS UC module shall:

a) if timer t-UC-2 is set, stop timer t-UC-2 ,

b) request the FIS HI module to invoke a FIS-cancel-update-contract indication with the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element,

c) create a FISDownlinkAPDU [FISCancelUpdateAccept] APDU based on the value of the *ContractNumber* APDU-element as received in the APDU,

d) request the LI module to send the APDU to the FIS-ground-ASE,

e) if there is no other FIS contract in place, start the t-inactivity timer, and

f) enter the UC-A-IDLE state.

2.4.5.3.7.9.2 If in the UC-A-PENDING state, the air FIS UC module shall:

a) stop timers t-UC-1 and t-UC-2 ,

b) request the FIS HI module to invoke a FIS-cancel-update-contract indication with the *FIS ContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element,

c) create a FISDownlinkAPDU [FISCancelUpdateAccept] APDU based on the value of the *ContractNumber* APDU-element as received in the APDU,

d) request the LI module to send the APDU to the FIS-ground-ASE,

e) if there is no other FIS contract in place, start the t-inactivity timer, and

f) enter the UC-A-IDLE state.

2.4.5.3.7.9.3 If in the UC-A-CANCEL state (i.e. collision of cancel-update-contract requests), the air FIS UC module shall:

a) create a FISDownlinkAPDU [FISCancelUpdateAccept] APDU based on the value of the *ContractNumber* APDU-element as received in the APDU,

b) request the LI module to send the APDU to the FIS-ground-ASE,

c) if there is no other FIS contract in place, start the t-inactivity timer, and

d) remain in the UC-A-CANCEL state.

2.4.5.3.7.10 Upon receipt of a [FISCancelUpdateAccept] APDU, then

2.4.5.3.7.10.1 If in the UC-A-CANCEL state, the air FIS UC module shall:

a) stop the t-UC-3 timer,

b) request the FIS HI module to invoke a FIS-cancel-update-contract confirmation with the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

c) enter the UC-A-IDLE state.

2.4.5.3.7.11 Upon receipt on a request from the AB module to stop operation, then

2.4.5.3.7.11.1 The air FIS UC module shall:

a) stop any timers, and

b) enter the UC-A-IDLE state.

2.4.5.3.7.12 Upon expiration of the t-UC-1 timer, then:

2.4.5.3.7.12.1 If in UC-A-PENDING state, the air FIS UC module shall:

a) stop timer t-UC-2,

b) request the AB module to abort with reason “timer expiration”, and

c) enter the UC-A-IDLE state.

2.4.5.3.7.13 Upon expiration of the t-UC-2 timer, then:

2.4.5.3.7.13.1 If in UC-A-ACTIVE state, the air FIS UC module shall:

a) request the AB module to abort with reason “timer expiration”, and

b) enter the UC-A-IDLE state.

2.4.5.3.7.14 Upon expiration of the t-UC-3 timer, then:

2.4.5.3.7.14.1 If in UC-A-CANCEL state, the air FIS UC module shall:

a) request the AB module to abort with reason “timer expiration”, and

b) enter the UC-A-IDLE state.

2.4.5.3.8 Ground FIS UC Module

Note.— The states defined for the ground FIS UC module are the following:

a) *UC-G-IDLE,*

b) *UC-G-PENDING,*

c) *UC-G-ACTIVE, and*

d) *UC-G-CANCEL.*

2.4.5.3.8.1 On initiation, the ground FIS UC module shall be in the UC-G-IDLE state.

2.4.5.3.8.2 Upon receipt of a [FISRequest] APDU containing an *updateContract* APDU-element, then

2.4.5.3.8.2.1 If in the UC-G-IDLE state, the ground FIS UC module shall:

a) request the FIS HI module to invoke a FIS-update-contract indication with the following parameters:

1) the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

2) the *FISContractDetails* parameter, containing the information which has been received as the *FISRequestData* APDU-element, and

b) enter the UC-G-PENDING state.

2.4.5.3.8.3 Upon receipt of a FIS-update-contract response with a *Result* parameter containing the abstract value “accepted”, then

2.4.5.3.8.3.1 If in the UC-G-PENDING state, the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISAccept] APDU with an *accept* APDU-element based on the value of the *FISContractNumber* and *FISInformation* parameters,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the UC-G-ACTIVE state.

2.4.5.3.8.4 Upon receipt of a FIS-update-contract response with a *Result* parameter containing the abstract value “positive acknowledgement”, then

2.4.5.3.8.4.1 If in the UC-G-PENDING state, the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISAccept] APDU with a *positiveAcknowledgement* APDU-element based on the value of the *FISContractNumber* parameter,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the UC-G-ACTIVE state.

2.4.5.3.8.5 Upon receipt of a FIS-update-contract response with a *Result* parameter containing the abstract value “rejected”, then

2.4.5.3.8.5.1 If in the UC-G-PENDING state, the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISReject] APDU based on the value of:

1) the *FISContractNumber* parameter and,

2) the *FISInformation* parameter if provided and if the *RejectReason* parameter contains the abstract value “update contract function not supported”, or

3) the *RejectReason* parameter if the *RejectReason* parameter does not contain the abstract value “update contract function not supported”,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the UC-G-IDLE state.

2.4.5.3.8.6 Upon receipt of a FIS-report request, then

2.4.5.3.8.6.1 If in the UC-G-ACTIVE state, the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISReport] APDU based on the value of the *FISContractNumber* and *FISInformation* parameters,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) remain in the UC-G-ACTIVE state.

2.4.5.3.8.7 Upon receipt of a FIS-cancel-update-contract request, then

2.4.5.3.8.7.1 If in the UC-G-ACTIVE state, the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISCancelUpdateContract] APDU based on the value of the *FISContractNumber* parameter,

b) start timer t-UC-3,

c) request the LI module to send the APDU to the FIS-air-ASE, and

d) enter the UC-G-CANCEL state.

2.4.5.3.8.7.2 If in the UC-G-PENDING state, the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISCancelUpdateContract] APDU based on the value of the *FISContractNumber* parameter,

b) start timer t-UC-3,

c) request the LI module to send the APDU to the FIS-air-ASE, and

d) enter the UC-G-CANCEL state.

2.4.5.3.8.8 Upon receipt of a [FISCancelUpdateContract] APDU, then

2.4.5.3.8.8.1 If in the UC-G-ACTIVE state, the ground FIS UC module shall:

a) request the FIS HI module to invoke a FIS-cancel-update-contract indication with the *ContractNumber* APDU-element as received in the APDU,

b) create a FISUplinkAPDU [FISCancelUpdateAccept] APDU based on the value of the *ContractNumber* APDU-element as received in the APDU,

c) request the LI module to send the APDU to the FIS-air-ASE, and

d) enter the UC-G-IDLE state.

2.4.5.3.8.8.2 If in the UC-G-PENDING state, the ground FIS UC module shall:

a) request the FIS HI module to invoke a FIS-cancel-update-contract indication with the *ContractNumber* APDU-element as received in the APDU,

b) create a FISUplinkAPDU [FISCancelUpdateAccept] APDU based on the value of the *ContractNumber* APDU-element as received in the APDU,

c) request the LI module to send the APDU to the FIS-air-ASE, and

d) enter the UC-G-IDLE state.

2.4.5.3.8.8.3 If in the UC-G-CANCEL state (i.e. collision of cancel-contract-update requests), the ground FIS UC module shall:

a) create a FISUplinkAPDU [FISCancelUpdateAccept] APDU based on the *ContractNumber* APDU-element as received in the APDU,

b) request the LI module to send the APDU to the FIS-air-ASE, and

c) enter the UC-G-IDLE state.

2.4.5.3.8.9 Upon receipt of a [FISCancelUpdateAccept] APDU, then

2.4.5.3.8.9.1 If in the UC-G-CANCEL state, the ground FIS UC module shall:

a) stop timer t-UC-3,

b) request the FIS HI module to invoke a FIS-cancel-update-contract confirmation with the *FISContractNumber* parameter containing the information which has been received as the *ContractNumber* APDU-element, and

c) enter the UC-G-IDLE state.

2.4.5.3.8.10 Upon receipt of a request from the AB module to stop operation, then

2.4.5.3.8.10.1 The ground FIS UC module shall enter the UC-G-IDLE state.

2.4.5.3.8.11 Upon expiration of the t-UC-3 timer, then:

2.4.5.3.8.11.1 If in the UC-G-CANCEL state, the ground FIS UC module shall:

a) request the AB module to abort with reason “timer expiration”, and

b) enter the UC-G-IDLE state.

2.4.5.3.9 Ground and Air AB Module

Note.— All statements in 2.4.5.3.9 apply to both the FIS ground AB module and the FIS air AB module.

2.4.5.3.9.1 Upon receipt of a request to abort from the HI, LI, DC or UC modules, the AB module shall:

a) request DC and UC modules to stop operation,

b) request the FIS HI module to invoke a FIS-provider-abort indication primitive with the *Reason* parameter set to value supplied by the module requesting the abort,

c) if the AB module is an air FIS module, create a FISDownlinkAPDU [FISAbort] APDU, with the *FISProtocolErrorDiag* APDU-element set to the value supplied by the module requesting the abort,

d) if the AB module is a ground FIS module, create a FISUplinkAPDU [FISAbort] APDU, with the *FISProtocolErrorDiag* APDU-element set to the value supplied by the module requesting the abort, and

e) request the LI module to send a D-ABORT request with the *originator* parameter set to the abstract value "provider".

2.4.5.3.9.2 Upon receipt of a HI request to send a FIS-user-abort request, the AB module shall:

a) request DC and UC modules to stop operation, and

b) request the LI module to send a D-ABORT request with the *originator* parameter set to the abstract value "user".

2.4.5.3.9.3 Upon receipt of a LI request to deliver a D-ABORT indication, the AB module shall:

a) request DC and UC modules to stop operation,

b) if the *originator* parameter contains the abstract value "user", request the FIS HI module to invoke a FIS-user-abort indication primitive, and

c) if the *originator* parameter contains the abstract value "provider", request the FIS HI module to invoke a FIS-provider-abort indication primitive with the *FISProtocolErrorDiag* APDU-element of the received APDU as *Reason* parameter.

2.4.5.3.9.4 Upon receipt of a LI request to deliver a D-P-ABORT indication, the AB module shall:

a) request DC and UC modules to stop operation, and

b) request the FIS HI module to invoke a FIS-provider-abort indication with the abstract value "communication system failure" as *Reason* parameter.

2.4.5.3.10 Air Cancel (CL) Module

2.4.5.3.10.1 Upon receipt of a HI request to send a FIS-cancel-contracts request, if there are DC and UC modules handling contracts of the types specified in the *FISServiceType* parameter, the CL module shall:

a) request these DC and UC modules to stop operation for the types of contracts concerned by the cancellation as specified in the *FISServiceType* parameter,

b) create a FISDownlinkAPDU [FISCancelContracts] APDU based on the value of the *FISServiceType* parameter,

c) request the LI module to send the APDU to the peer FIS-ASE, and

d) start the t-CL-1 timer.

2.4.5.3.10.2 Upon receipt of [FISCancelContractsAccept] APDU and the *FISServiceType* APDU-element matches exactly the *FISServiceType* APDU-element of the [FISCancelContracts] APDU sent previously, the CL module shall:

a) stop the t-CL-1 timer,

b) request the FIS HI module to invoke a FIS-cancel-contracts confirmation primitive with the types of contracts concerned by the cancellation specified in the [FISCancelContractsAccept] APDU as *FISServiceType* parameter, and

c) start the t-inactivity timer if there is no other FIS contract in place.

2.4.5.3.10.3 Upon expiration of the timer t-CL-1, the CL module shall request the AB module to abort with reason "timer expiration".

2.4.5.3.11 Ground Cancel (CL) Module

2.4.5.3.11.1 Upon receipt of [FISCancelContracts] APDU, the CL module shall:

a) request DC and UC modules to stop operation for the types of contracts concerned by the cancellation as specified in the [FISCancelContracts] APDU,

b) request the FIS HI module to invoke a FIS-cancel-contracts indication primitive with the type(s) of contracts concerned by the cancellation specified in the [FISCancelContracts] APDU as *FISServiceType* parameter,

c) create a FISUplinkAPDU [FISCancelContractsAccept] APDU based on the value of the *FISServiceType* parameter, and

d) request the LI module to send the APDU to the peer FIS-ASE.

2.4.5.3.12 Ground and Air Low Interface Module

Note 1.— All statements in 2.4.5.3.12 apply to both the FIS ground LI module and the FIS air LI module.

Note 2.— Table 2.4.5-4 specifies the type of the APDUs the LI module can send to the peer FIS-ASE.

Table 2.4.5-4. Types of the FIS-ASE APDUs

APDU	**Type**
FISRequest	initial
FISAccept	normal
FISReject	normal

APDU	**Type**
FISReport	normal
FISCancelUpdateContract	normal
FISCancelUpdateAccept	normal
FISCancelContracts	normal
FISCancelContractsAccept	normal

Note 3.— The states defined for the LI module are the following:

a) LI-IDLE,

b) LI-START-I (initiator of the D-START req),

c) LI-START-R (receptor of the D-START ind),

d) LI-DIALOGUE, and

e) LI-END-I (initiator of the D-END req).

2.4.5.3.12.1 On initiation, the LI module shall be in the LI-IDLE state.

2.4.5.3.12.2 On receipt of a request to send an APDU, the FIS LI module shall determine the APDU type based on the Table 2.4.5-4.

2.4.5.3.12.3 When requested by a DC or UC module to send an initial APDU to the peer ASE, then

2.4.5.3.12.3.1 If in the LI-IDLE state, the LI module shall:

a) invoke the D-START request primitive with the following parameters:

1) the ICAO Facility Designation provided by the DC or UC module as the *CalledPeerId* parameter,

2) the *QualityOf Service* parameter composed as follows:

i) the Routing Class based on the value of the *ClassOfCommunicationService* parameter if provided by the DC or UC module,

ii) the Application Service Priority as defined in 2.4.6.2.2.1, and

iii) the RER as defined in 2.4.6.2.2.2, and

3) the APDU as *UserData* parameter, and

b) enter the LI-START-I state.

2.4.5.3.12.3.2 If in the LI-DIALOGUE state, the LI module shall:

a) invoke the D-DATA request primitive with the APDU as *UserData* parameter, and

b) remain in the LI-DIALOGUE state.

2.4.5.3.12.4 When requested by a DC or UC module to send a normal APDU to the peer ASE, then

2.4.5.3.12.4.1 If in the LI-DIALOGUE state, the LI module shall:

a) invoke the D-DATA request primitive with the APDU as *UserData* parameter, and

b) remain in the LI-DIALOGUE state.

2.4.5.3.12.4.2 If in the LI-START-R state, the LI module shall:

a) invoke the D-START response primitive with the following parameters:

1) the abstract value “accepted” as the *Result* parameter, and

2) the APDU as the *UserData* parameter, and

b) enter the LI-DIALOGUE state.

2.4.5.3.12.5 When requested by the AB module to send a D-ABORT request, then

2.4.5.3.12.5.1 If not in the LI-IDLE state, the LI module shall:

a) invoke the D-ABORT request primitive with the APDU as *UserData* parameter and the value supplied by the AB module as *Originator* parameter, and

b) enter the LI-IDLE state.

2.4.5.3.12.5.2 If in the LI-IDLE state, the LI module shall ignore the request.

2.4.5.3.12.6 Upon receipt of a D-START indication with the *UserData* parameter containing a FISDownlinkAPDU [FISRequest] APDU and the D-START Indication application service priority parameter has the abstract value “Aeronautical Information Service messages” and the RER Quality of Service Parameter has the abstract value “low”, then :

2.4.5.3.12.6.1 If in the LI-IDLE state, the LI module shall:

a) pass the APDU to the DC module if the APDU contains an *demandContract* APDU-element,

b) pass the APDU to the UC module if the APDU contains an *updateContract* APDU-element, and

c) enter the LI-START-R state.

2.4.5.3.12.7 Upon receipt of a D-START confirmation with a *Result* parameter containing the abstract value "accepted" and with a *UserData* parameter containing a FISUplinkAPDU [FISAccept] or [FISReject] APDU, then

2.4.5.3.12.7.1 If in the LI-START-I state, the LI module shall:

a) identify the module from the *ContractNumber* parameter of the received APDU,

b) pass the APDU to the relevant DC or UC module, and

c) enter the LI-DIALOGUE state.

2.4.5.3.12.8 Upon receipt of a D-START confirmation with the *RejectSource* parameter containing the abstract value "DS provider", then:

2.4.5.3.12.8.1 If in the LI-START-I state, the LI module shall:

a) request the FIS HI module to invoke a FIS-provider-abort indication with the abstract value "cannot establish contact with the peer" as *Reason* parameter, and

b) enter the LI-IDLE state.

2.4.5.3.12.9 Upon receipt of a D-START confirmation with the *RejectSource* parameter containing the abstract value "DS user", then:

2.4.5.3.12.9.1 If in the LI-START-I state, the LI module shall:

a) request the FIS HI module to invoke a FIS-provider-abort indication with the abstract value "contact refused by the peer" as *Reason* parameter, and

b) enter the LI-IDLE state.

2.4.5.3.12.10 Upon receipt of a D-DATA indication with a *UserData* parameter containing a valid APDU (i.e. any APDU except [FISAbort] APDU), then

2.4.5.3.12.10.1 If in the LI-DIALOGUE or LI-END-I state, the LI module shall:

a) identify the module:

1) the DC module if the [FISRequest] APDU contains an *demandContract* APDU-element,

2) the UC module if the [FISRequest] APDU contains an *updateContract* APDU-element,

3) the DC or UC module based on the *ContractNumber* APDU-element of the received APDU, if the APDU is one of the following: [FISAccept], [FISReject], [FISCancelUpdateContract], [FISCancelUpdateAccept] or [FISReport], or

4) the CL module if the APDU is [FISCancelContracts] or [FISCancelContractsAccept],

b) pass the APDU to that module, and

c) remain in the same state.

2.4.5.3.12.11 Upon receipt of a D-END indication, then

2.4.5.3.12.11.1 If in the LI-DIALOGUE state, the LI module shall:

a) invoke the D-END response primitive with the abstract value “accepted” as *Result* parameter, and

b) enter the LI-IDLE state.

2.4.5.3.12.12 Upon receipt of a D-END confirmation with a *Result* parameter containing the abstract value “accepted”, then

2.4.5.3.12.12.1 If in the LI-END-I state, the LI module shall enter the LI-IDLE state.

2.4.5.3.12.13 Upon receipt of a D-END confirmation with a *Result* parameter containing the abstract value “rejected”, then:

2.4.5.3.12.13.1 If in the LI-END-I state, the LI module shall:

a) request the AB module to abort with reason “dialogue-end-not-supported”, and

b) remain in the LI-END-I state.

2.4.5.3.12.14 Upon receipt of a D-ABORT indication with the *UserData* parameter, the LI module shall:

a) forward the primitive to the AB Module, and

b) enter the LI-IDLE state.

2.4.5.3.12.15 Upon receipt of a D-P-ABORT indication, the LI module shall:

a) forward the primitive to the AB module, and

b) enter the LI-IDLE state.

2.4.5.3.12.16 Upon receipt of an indication that the t-inactivity timer has expired, then

2.4.5.3.12.16.1 If in LI-DIALOGUE state, the LI module shall:

a) start the timer t-LI-1,

b) invoke the D-END request, and

c) enter the LI-END-I state.

2.4.5.3.12.17 Upon receipt of an indication that the LI-1 timer has expired, then

2.4.5.3.12.17.1 If in LI-END-I state, the LI module shall:

a) invoke the D-ABORT request primitive with the abstract value "user" as the *Originator* parameter, and

b) enter the LI-IDLE state.

2.4.5.4 Exception Handling

2.4.5.4.1 Unrecoverable Internal Error

2.4.5.4.1.1 **Recommendation.—** *When any module has an unrecoverable system error, the module should:*

a) request the AB module to abort with reason "unrecoverable internal error", and

b) remains in its current state.

2.4.5.4.2 Protocol Error

2.4.5.4.2.1 When the LI module detects that the *UserData* parameter of a dialogue service indication or confirmation does not contain an APDU and the service is not D-END, the LI module shall:

a) request the AB module to abort with the reason "protocol error", and

b) remain in its current state.

2.4.5.4.3 Sequence Error

2.4.5.4.3.1 When an APDU is passed to any module except the LI and the HI modules for which there are no instructions in 2.4.5.3 (i.e. the PDU has arrived out of sequence), that module shall:

a) request the AB module to abort with the reason “sequence error”, and

b) remains in its current state.

2.4.5.4.4 Decoding Error

2.4.5.4.4.1 When the LI module fails to decode an APDU, the LI module shall:

a) request the AB module to abort with the reason “decoding error”, and

b) remain in its current state.

2.4.5.4.5 Invalid FIS Contract Number

2.4.5.4.5.1 When the LI module detects that the *ContractNumber* APDU-element in the decoded APDU is invalid (i.e. the *ContractNumber* APDU-element in a non initial APDU is not already in use or the *ContractNumber* APDU-element in an initial APDU is already in use), the LI module shall:

a) request the AB module to abort with the reason “invalid contract number”, and

b) remain in its current state.

2.4.5.4.6 D-START Indication Quality of Service Parameter Not As Expected

2.4.5.4.6.1 When the LI module detects that the *QualityOfService* parameter of a D-START indication primitive does not contain the abstract value “Aeronautical Information Service messages” as application service priority or does not contain the abstract value “low” as RER, the LI module shall:

a) request the AB module to abort with the reason “invalid QOS parameter”, and

b) remain in its current state.

2.4.5.5 State Tables

2.4.5.5.1 Priority

2.4.5.5.1.1 If the state tables for the modules of the FIS-air-ASE and the FIS-ground-ASE.shown below conflict with textual statements made elsewhere in this document, the textual statements shall take precedence.

Note 1.— The following notation conventions apply for the following state tables :

a) Module States are identified as follows : <module name>-<location(A/G)>-<state>.

b) Events between a source module and a recipient module are identified as shown in the following figure:

Note 2.— In the following state tables, the statement "cannot occur" means that if the implementation conforms to the SARPs, it is impossible for this event to occur. If the event does occur, this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE aborts with the error "unrecoverable internal error".

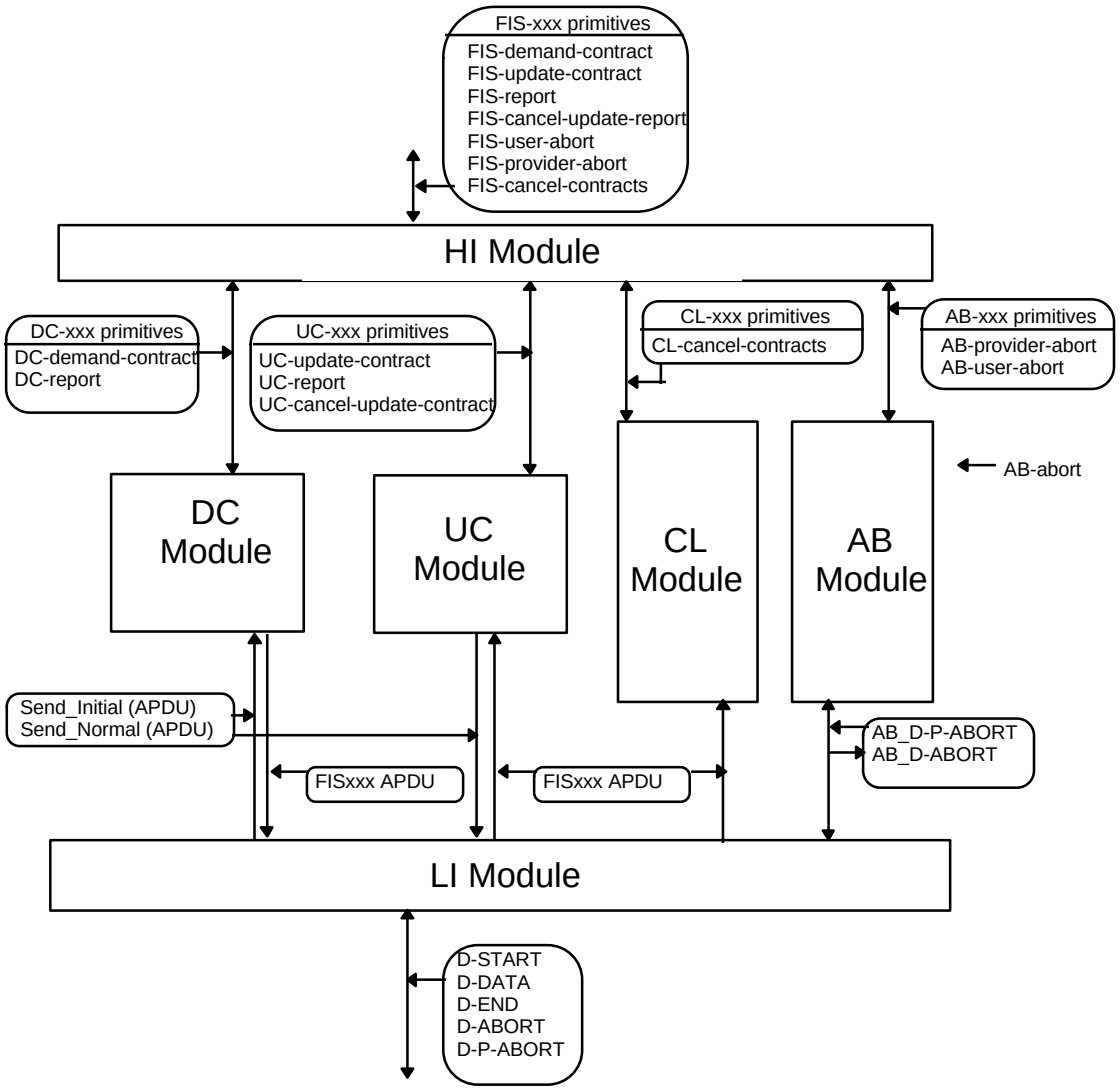

Figure 2.4.5-19. Functional Model for FIS ASE State Tables

Note 3.— In the following state tables, the statement "not permitted" means that the implementation must prevent this event from occurring through some local means. If the event does occur this implies that there is an error in the implementation. If such a situation is detected, it is suggested that the ASE performs a local rejection of the request rather than aborting the dialogue.

Note 4.— As the HI Module provides the ASE-users with a straightforward pass-through service to the DC, UC, CL and AB Modules, state tables related to the HI Modules are not described. FIS-xxx request and response primitives are mapped onto DC-xxx, UC-xxx, CL-xxx or AB-xxx request and response primitives; DC-xxx, UC-xxx, CL-xxx or AB-xxx indication and confirmation primitives are mapped onto FIS-xxx indication and confirmation primitives, with the following exception: when AB-user-abort indication or AB-provider-abort indication are received by the HI Module from the AB Module, the corresponding FIS-user-abort indication or FIS-provider-abort indication is only generated to the FIS-user when the FIS-user is active.

Table 2.4.5-6/a. Air FIS DC Module State Table

State ⇒ Event ⇓	DC-A-IDLE (Initial State)	DC-A-PENDING	DC-A-ACTIVE
Primitive Requests and Responses from FIS HI module			
DC-demand-contract req	LI-Send-Initial [FISRequest-DemandContract] APDU Start t-DC-1, t-DC-2 ➜DC-A-PENDING	not permitted	not permitted
FIS APDUs from LI module			
[FISReport] APDU	cannot occur	AB-abort (“sequence error”) ➜DC-A-IDLE	stop t-DC-2 if last FIScontract start t-INACTIVITY DC-report ind ➜DC-A-IDLE
[FISAccept-accept] APDU	cannot occur	stop t-DC-1, t-DC-2 if last FISContract start t-INACTIVITY DC-demand-contract cnf ➜DC-A-IDLE	AB-abort (“sequence error”) ➜DC-A-IDLE
[FISAccept-positive acknowledgement] APDU	cannot occur	stop t-DC-1 DC-demand-contract cnf ➜DC-A-ACTIVE	AB-abort (“sequence error”) ➜DC-A-IDLE
[FISReject] APDU	cannot occur	stop t-DC-1, t-DC-2 if last FIScontract start t-INACTIVITY DC-demand-contract cnf ➜DC-A-IDLE	AB-abort (“sequence error”) ➜DC-A-IDLE
Requests from AB module			
DC-stop-operation	➜DC-A-IDLE	stop t-DC-1, t-DC-2 ➜DC-A-IDLE	stop t-DC-2 ➜DC-A-IDLE

Table 2.4.5-6/b. Air FIS DC Module State Table

State ⇒ Event ⇓	DC-A-IDLE (Initial State)	DC-A-PENDING	DC-A-ACTIVE
Timer expiration			
t-DC-2	cannot occur	cannot occur	AB-abort (“timer expiration”) ➜DC-A-IDLE
t-DC-1	cannot occur	Stop t-DC-2 AB-abort (“timer expiration”) ➜DC-A-IDLE	cannot occur

Table 2.4.5-7. Ground FIS DC Module State Table

State ⇒ Event ⇓	DC-G-IDLE (Initial State)	DC-G-PENDING	DC-G-ACTIVE
Primitive Requests and Responses from HI Module			
DC-demand-contract rsp (accepted)	not permitted	LI-Send-Normal [FISAccept-accept] APDU ➜DC-G-IDLE	not permitted
DC-demand-contract rsp (rejected)	not permitted	LI-Send-Normal [[FISReject] APDU ➜DC-G-IDLE	not permitted
DC-demand-contract rsp (positive acknowledgement)	not permitted	LI-Send-Normal [FISAccept-positive-ack] APDU ➜DC-G-ACTIVE	not permitted
DC-report req	not permitted	not permitted	LI-Send-Normal [FISReport] APDU ➜DC-G-IDLE
APDU from LI Module			
[FISRequest-DemandContract] APDU	DC-demand-contract ind ➜DC-G-PENDING	AB-abort (“sequence error”) ➜DC-G-IDLE	AB-abort (“sequence error”) ➜DC-G-IDLE
APDU from AB Module			
DC-stop-operation	➜DC-G-IDLE	➜DC-G-IDLE	➜DC-G-IDLE

Table 2.4.5-8/a. Air FIS UC Module State Table

State ⇒ Event ⇓	UC-A-IDLE (Initial State)	UC-A-PENDING	UC-A-ACTIVE	UC-A-CANCEL
Primitive Requests and Responses from HI				
UC-update-contract req	LI-Send-Initial [FIS-Request-Update Contract] APDU Start t-UC-1, t-UC-2 ➜UC-A-PENDING	not permitted	not permitted	not permitted
UC-cancel-update-contract req	not permitted	if dialogue not established not permitted else stop t-UC-1, t-UC-2 LI-Send-Normal [FISCancelUpdate Contract] APDU start T-UC-3 CANCELFROM PENDING = TRUE ➜UC-A-CANCEL	if set, stop t-UC-2 LI-Send-Normal [FISCancelUpdate Contract] APDU start t-UC-3 ➜UC-A-CANCEL	not permitted
FIS uplink PDUs from LI				
[FISAccept-accept] APDU	cannot occur	stop t-UC-1, t-UC-2 UC-update-contract cnf ➜UC-A-ACTIVE	AB-abort (“sequence error”) ➜UC-A-IDLE	If CANCELFROM PENDING = TRUE stop t-UC-1, t-UC-2 CANCELFROM PENDING = FALSE ➜UC-A-CANCEL
[FISAccept-positive acknowledgement] APDU	cannot occur	stop t-UC-1 UC-update-contract cnf ➜UC-A-ACTIVE	AB-abort (“sequence error”) ➜UC-A-IDLE	If CANCELFROM PENDING = TRUE stop t-UC-1, t-UC-2 CANCELFROM PENDING = FALSE ➜UC-A-CANCEL

Table 2.4.5-8/b. Air FIS UC Module State Table

State ⇒ Event ⇓	UC-A-IDLE (Initial State)	UC-A-PENDING	UC-A-ACTIVE	UC-A-CANCEL
[FISReject] APDU	cannot occur	stop t-UC-1, t-UC-2 if last FIScontract start t-INACTIVITY UC-update-contract cnf ➜UC-A-IDLE	AB-abort ("sequence error") ➜UC-A-IDLE	If CANCELFROM PENDING = TRUE stop t-UC-1, t-UC-2 CANCELFROM PENDING = FALSE UC-cancel-update-contract cnf ➜UC-A-IDLE else AB-abort ("sequence error") ➜UC-A-IDLE
[FISReport] APDU	cannot occur	AB-abort ("sequence error") ➜UC-A-IDLE	If set, stop t-UC-2 UC-report ind ➜UC-A-ACTIVE	If CANCELFROM PENDING = FALSE ➜UC-A-CANCEL else AB-abort ("sequence error") ➜UC-A-IDLE
[FISCancelUpdateContract] APDU	cannot occur	UC-cancel-update-contract ind LI-Send-Normal [FISCancelUpdate Accept] APDU if last FIS Contract start t-inactivity ➜UC-A-IDLE	If set, stop t-UC-2 UC-cancel-update-contract ind LI-Send-Normal [FISCancelUpdate Accept] APDU if last FIS Contract start t-inactivity ➜UC-A-IDLE	/* collision */ LI-Send-Normal [FISCancelUpdateAccept] APDU if last FIS Contract start t-inactivity ➜UC-A-CANCEL
[FISCancelUpdateAccept] APDU	cannot occur	AB-abort ("sequence error") ➜UC-A-IDLE	AB-abort ("sequence error") ➜UC-A-IDLE	stop t-UC-3 UC-cancel-update-contract cnf ➜UC-A-IDLE
Request from AB module				
UC-stop-operation	➜UC-A-IDLE	stop t-UC-1, t-UC-2 ➜UC-A-IDLE	stop t-UC-2 ➜UC-A-IDLE	➜UC-A-IDLE
Timer Expiration				
t-UC-1	cannot occur	Stop t-UC-2 AB-abort ("timer expiration") ➜UC-A-IDLE	cannot occur	cannot occur
t-UC-2	cannot occur	cannot occur	AB-abort ("timer expiration") ➜UC-A-IDLE	cannot occur

State ⇒ Event ⇓	UC-A-IDLE (Initial State)	UC-A-PENDING	UC-A-ACTIVE	UC-A-CANCEL
t-UC-3	cannot occur	cannot occur	cannot occur	AB-abort ("timer expiration") ➜UC-A-IDLE

Table 2.4.5-9/a. Ground FIS UC Module State Table

State ⇒ Event ⇓	UC-G-IDLE (Initial State)	UC-G-PENDING	UC-G-ACTIVE	UC-G-CANCEL
Primitive Requests and Responses from HI				
UC-update-contract rsp (accepted)	not permitted	LI-Send-Normal [FISAccept-accept] APDU ➜UC-G-ACTIVE	not permitted	not permitted
UC-update-contract rsp (rejected)	not permitted	LI-Send-Normal [FISReject] APDU ➜UC-G-IDLE	not permitted	not permitted
UC-update-contract rsp (processing)	not permitted	LI-Send-Normal [FISAccept-processing] APDU ➜UC-G-ACTIVE	not permitted	not permitted
UC-report req	not permitted	not permitted	LI-Send-Normal [FISReport] APDU ➜UC-G-ACTIVE	not permitted
UC-cancel-update-contract req	not permitted	LI-Send-Normal [FISCancelUpdate Contract] APDU start t-UC-3 ➜UC-G-CANCEL	LI-Send-Normal [FISCancelUpdate Contract] APDU start t-UC-3 ➜UC-G-CANCEL	not permitted

Table 2.4.5-9/b. Ground FIS UC Module State Table

State ⇒ Event ⇓	UC-G-IDLE (Initial State)	UC-G-PENDING	UC-G-ACTIVE	UC-G-CANCEL
FIS downlink PDUs from LI				
[FISRequest-UpdateContract] APDU	UC-update-contract ind ➜UC-G-PENDING	AB-abort (“protocol error”) ➜ UC-G-IDLE	AB-abort (“protocol error”) ➜ UC-G-IDLE	AB-abort (“protocol error”) ➜ UC-G-IDLE
[FISCancelUpdateContract] APDU	cannot occur	UC-cancel-update-contract ind LI-Send-Normal [FISCancelUpdate Accept] APDU ➜ UC-G-IDLE	UC-cancel-update-contract ind LI-Send-Normal [FISCancelUpdate Accept] APDU ➜ UC-G-IDLE	/* collision */ LI-Send-Normal [FISCancelUpdate Accept] APDU ➜ UC-G-IDLE
[FISCancelUpdateAccept] APDU	cannot occur	AB-abort (“protocol error”) ➜ UC-G-IDLE	AB-abort (“protocol error”) ➜ UC-G-IDLE	stop t-UC-3 FIS-cancel-update-contract cnf ➜UC-G-IDLE
Request from AB module				
UC-stop-operation	➜ UC-G-IDLE	➜ UC-G-IDLE	➜ UC-G-IDLE	➜ UC-G-IDLE
Timer Expiration				
t-UC-3	cannot occur	cannot occur	cannot occur	AB-abort (“timer expiration”) ➜ UC-G-IDLE

Table 2.4.5-10/a. Air and ground FIS LI Modules State Table

State ⇒ Event ⇓	LI-IDLE (Initial State)	LI-START-I	LI-DIALOGUE	LI-START-R	LI-END-I
Primitive Requests and Responses from DC, UC or AB Module					
LI-Send-Initial (APDU)	D-START req ➜ LI-START- I	not permitted	D-DATA req	not permitted /* only air-initiated */	not permitted
LI-Send-Normal (APDU)	not permitted	not permitted	D-DATA req	D-START rsp ➜ LI-DIALOGUE	not permitted
Primitive Requests from AB Module					
AB_D-ABORT req	➜ LI-IDLE	D-ABORT req ➜ LI-IDLE	D-ABORT req ➜ LI-IDLE	D-ABORT req ➜ LI-IDLE	D-ABORT req ➜ LI-IDLE
Timer Expiration					
t-LI-1	cannot occur	cannot occur	cannot occur	cannot occur	D-ABORT req ➜LI-IDLE
t-INACTIVITY	cannot occur	cannot occur	start t-LI-1 D-END req ➜ LI-END-I	cannot occur	cannot occur

Table 2.4.5-10/b. Air and ground FIS LI Modules State Table

State ⇒ Event ⇓	LI-IDLE (Initial State)	LI-START-I	LI-DIALOGUE	LI-START-R	LI-END-I
Dialogue Service primitives from DSP					
D-START ind	Pass APDU ➜ LI-START- R	cannot occur	cannot occur	cannot occur	cannot occur
D-START cnf positive (result=accepted)	cannot occur	Pass APDU ➜ LI-DIALOGUE	cannot occur	cannot occur	cannot occur
D-START cnf negative (result=rejectedtransient & source=DS provider)	cannot occur	AB-dialogue-abort (“cannot establish contact with the peer”) ➜ LI-IDLE	cannot occur	cannot occur	cannot occur
D-START cnf negative (result=rejectedtransient & source=DS user)	cannot occur	AB-dialogue-abort (“contact refused by the peer”) ➜ LI-IDLE	cannot occur	cannot occur	cannot occur
D-DATA ind	cannot occur	cannot occur	Pass APDU	cannot occur	Pass APDU
D-END ind	cannot occur	cannot occur	D-END rsp positive ➜ LI-DLE	cannot occur	cannot occur
D-END cnf positive	cannot occur	cannot occur	cannot occur	cannot occur	➜ LI-IDLE
D-END cnf negative	cannot occur	cannot occur	cannot occur	cannot occur	AB-dialogue-abort (“dialogue end not supported”) ➜ LI-END-I
D-ABORT ind	cannot occur	AB_D-ABORT ind ➜ LI-IDLE	AB_D-ABORT ind ➜ LI-IDLE	AB_D-ABORT ind ➜ LI-IDLE	AB_D-ABORT ind ➜ LI-IDLE
D-P-ABORT ind	cannot occur	AB_D-P-ABORT ind ➜ LI-DLE	AB_D-P-ABORT ind ➜ LI-DLE	AB_D-P-ABORT ind ➜ LI-DLE	AB_D-P-ABORT ind ➜ LI-DLE

Table 2.4.5-11. Air and Ground FIS AB Modules State Table

AB module state ⇒ Event ⇓	AB-IDLE
Primitive Requests from DC and UC Module	
AB-abort (reason)	UC-stop-operation (all FIS services) DC-stop-operation AB-provider-abort ind (reason) AB-D-ABORT req (“provider ” [FISAbort] APDU)
Primitive Indications and APDUs from LI Module	
AB_D-P-ABORT ind (reason)	UC-stop-operation (all FIS services) DC-stop-operation (all FIS services) AB-provider-abort ind (reason)
AB_D-ABORT ind (« provider »,APDU)	UC-stop-operation (all FIS services) DC-stop-operation (all FIS services) AB-provider-abort ind (APDU.AbortReason)
AB_D-ABORT ind (« user »)	UC-stop-operation (all FIS services) DC-stop-operation (all FIS services) AB-user-abort ind ()
Primitive Requests from HI Module	
AB-user-abort req	UC-stop-operation (all FIS services) DC-stop-operation (all FIS services) AB-D-ABORT req (« user »)

Table 2.4.5-12. Air FIS CL Module State Table

CL module state ⇒	CL-IDLE
Event ⇓	
Primitive Request from the FIS HI module	
CL-cancel-contracts req (types of FIS service)	if UC or DC exist with these types of FIS service UC-stop-operation (type of FIS service) DC-stop-operation (type of FIS service) LI-Send-Normal [FISCancelContracts] APDU start t-CL-1
APDUs from LI Module	
[FISCancelContractsAccep t] APDU	stop t-CL-1 CL-cancel-contracts cnf if last FIS Contract, start t-inactivity
Timer Expiration	
t-CL-1	AB-abort (« timer expiration »)

Table 2.4.5-13. Ground FIS CL Module State Table

CL module state ⇒	CL-IDLE
Event ⇓	
APDUs from LI Module	
[FISCancelContracts] APDU	UC-stop-operation (type of FIS service) DC-stop-operation (type of FIS service) CL-cancel-contracts ind LI-Send-Normal [FISCancelContractsAccept] APDU

2.4.6 COMMUNICATION REQUIREMENTS

2.4.6.1 Encoding Rules

2.4.6.1.1 The FIS application shall use PER as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in 2.4.4.

2.4.6.2 Dialogue Service Requirements

2.4.6.2.1 Primitives Requirements

2.4.6.2.1.1 Where dialogue service primitives, that is D-START, D-END, D-ABORT, D-P-ABORT, and D-DATA are described as being invoked in 2.4.5, the FIS-ground-ASE and the FIS-air-ASE shall exhibit external behavior consistent with the dialogue service, as described in 4.2, having been implemented and its primitives invoked.

2.4.6.2.2 Quality Of Service Requirements

2.4.6.2.2.1 The application service priority for ATIS shall have the abstract value of “Aeronautical Information Service messages”.

2.4.6.2.2.2 The Residual Error Rate (RER) Quality Of Service parameter of the D-START request shall be set to the abstract value of “low”.

2.4.6.2.2.3 The FIS-ASE shall map the FIS-service Class of Communication Service abstract value to the ATSC routing class abstract value part of the D-START *Quality Of Service* parameter as presented in Table 4.2.6-1.

Table 4.2.6-1. Mapping between Class of Communication and Routing Class Abstract Values

Class of Communication Abstract Value	Routing Class Abstract Value
A	Traffic follows Class A ATSC route(s)
B	Traffic follows Class B ATSC route(s)
C	Traffic follows Class C ATSC route(s)
D	Traffic follows Class D ATSC route(s)
E	Traffic follows Class E ATSC route(s)
F	Traffic follows Class F ATSC route(s)
G	Traffic follows Class G ATSC route(s)
H	Traffic follows Class H ATSC route(s)

Note.— ATSC values are defined in 1.3.

2.4.7 FIS USER REQUIREMENTS

2.4.7.1 Introduction

Note.— Requirements imposed on FIS-users interfacing with the FIS-ASEs are presented in 2.4.7.

2.4.7.2 General

2.4.7.2.1 General Requirements

Note 1.— When a FIS-air-user invokes FIS-demand-contract request or FIS-update-contract request and requires a particular class of communication service, it sets the ClassOfCommunicationService parameter to be the class of communication service it requires.

Note 2.— When a FIS-air-user invokes FIS-demand-contract request or FIS-update-contract request and does not provide the ClassOfCommunicationService parameter, this indicates no routing preference.

Note 3.— When a FIS-air-user specifies the ClassOfCommunicationService parameter and there is a FIS contract in place, the ClassOfCommunicationService parameter is ignored.

2.4.7.2.2 Response Time Requirements

2.4.7.2.2.1 **Recommendation.—** *Upon receipt of a FIS-ASE service indication that requires a response, the FIS-user should invoke the FIS-ASE service response within 1 second.*

2.4.7.2.3 Error Handling Requirements

2.4.7.2.3.1 If a FIS-user has an unrecoverable system error it shall:

a) cease the operation of all contracts with peer systems which are affected by the error, and

b) for each affected peer system, invoke FIS-user-abort request.

2.4.7.2.3.2 If a FIS-user receives a FIS-provider-abort indication or a FIS-user-abort indication, it shall cease operation of all FIS contracts with the peer system to which the indication is related.

2.4.7.2.4 Miscellaneous Air and Ground FIS Systems Requirements

2.4.7.2.4.1 With the exception of the FIS-user-abort and FIS-provider-abort, the FIS-user shall respond to FIS-ASE service indications and confirmations from the FIS-ASE in the order in which they are received.

2.4.7.2.5 Miscellaneous Ground FIS System Requirements

2.4.7.2.5.1 The FIS-ground-user shall be capable of detecting that the content of the FIS request is not valid or that the requested FIS information is not available.

2.4.7.2.5.2 The FIS-ground-user shall be capable of detecting that it can not continue to provide updates of the requested FIS information.

2.4.7.2.5.3 The FIS-ground-user shall be able to support multiple concurrent FIS contracts with a given aircraft.

2.4.7.2.6 ATIS Service Requirements

2.4.7.2.6.1 If only the arrival ATIS or only the departure ATIS is requested, and if the FIS-ground-user only provides combined ATIS, the FIS-ground-user shall send combined ATIS with a 'combined' indication.

2.4.7.2.6.2 If both arrival and departure ATIS are requested, the FIS-ground-user shall:

a) send a combined ATIS with a 'combined' indication, if the combined ATIS is available, or

b) send an arrival ATIS and a departure ATIS if both information are available but the combined ATIS is not available, or

c) send the available ATIS with an indication that the other is not available, if only one is available.

2.4.7.2.6.3 **Recommendation**.— *The ATIS fields should be presented in the following order:*

a) Airport identification,

b) Departure or arrival indicator,

c) ATIS Code,

d) Time of Observation, if present,

e) Approach Type, if present,

f) Runways In Use,

g) RunwayArresting System, if present,

h) RunwaySurfaceConditions, if present,

i) RunwayBraking Action, if present,

j) Holding Delay, if present,

k) Transition Level, if present,

l) Other Operational Information, if present,

m) Surface Wind,

n) Visibility,

o) RVR, if present,

p) Present Weather,

q) Cloud Sky and Cover,

r) Air Temperature,

s) Dew Point Temperature,

t) Altimeter Setting,

u) Significant Met Information,

v) Trend Type of Landing Forecast, if present, and

w) Specific ATIS Instruction, if present.

2.4.7.3 Establishment And Operation Of a FIS Demand Contract

Note 1.— When a FIS-air-user requires to establish a FIS demand contract with a FIS-ground-user it invokes FIS-demand-contract request.

Note 2.— Only the FIS-air-user is capable of initiating the FIS-demand-contract service.

Note 3.— If the FIS-air-user uses a value for the FISContractNumber parameter already in use, the FIS Demand Contract request will be rejected.

Note 4.— If the FIS-air-user invokes a second FIS-demand-contract request before the very first request (either a FIS-demand-contract or a FIS-update-contract request) has been confirmed, the FIS-demand-contract will be rejected.

2.4.7.3.1 The FIS-ground-user shall use the value received in the *FISContractNumber* parameter of the FIS-demand-contract indication for the *FISContractNumber* parameter of any ground-initiated request or response related to that FIS demand contract.

2.4.7.3.2 The same type of FIS information (i.e. “ATIS” for version 1 of the FIS-ASEs) shall be identified by the FIS-air-user and FIS-ground-user when invoking the FIS-demand-contract request and response primitives.

2.4.7.3.3 Upon receipt of a FIS-demand-contract indication, when the FIS-ground-user is able to accept the contract in full, then:

2.4.7.3.3.1 If the FIS-ground-user is not able to send the FIS report within the response time specified in 2.4.7.2.2., it shall:

a) invoke the FIS-demand-contract response with the *Result* parameter set to abstract value "positive acknowledgment", and

b) invoke the FIS-report request later.

2.4.7.3.3.2 **Recommendation.—** *If the FIS-ground-user is not able to send the FIS report within the response time, it should invoke the FIS-demand-contract response within the response time and then invoke the FIS-report request within 10 seconds.*

2.4.7.3.3.3 If the FIS-ground-user is able to send the FIS report within the response time specified in 2.4.7.2.2., it shall invoke the FIS-demand-contract response with the following parameters:

a) the Result parameter set to the abstract value "accepted", and

b) the FIS report in the *FIS Information* parameter.

2.4.7.3.4 Upon receipt of a FIS-demand-contract indication, if the FIS-ground-user cannot comply with the demand contract request, it shall reject the contract by invoking a FIS-demand-contract response with the following parameters:

a) the abstract value "rejected" as *Result* parameter, and

b) one of the following abstract values "cannot comply", "FIS service not available", "error detected in the FIS request" or "undefined", as *RejectReason* parameter.

2.4.7.3.5 The FIS-air-user shall be allowed to reuse the value of a *FISContractNumber* used in a FIS-demand-contract request for a new FIS contract:

a) upon receipt of a FIS-demand-contract confirmation with a *Result* parameter containing the abstract value "rejected" or "accepted",

b) upon receipt of a FIS-report indication, a FIS-user-abort indication or a FIS-provider-abort indication, or

c) after invocation of a FIS-user-abort request.

2.4.7.4 Establishment and Operation of a FIS Update Contract

Note 1.— When an FIS-air-user requires to establish a FIS update contract with an FIS-ground-user it invokes the FIS-update-contract request.

Note 2.— Only the FIS-air-user is capable of initiating the FIS-update-contact service.

Note 3.— If the FIS-air-user uses a value for the FISContractNumber parameter already in use, the FIS Update Contract request will be rejected.

Note 4.— If the FIS-air-user invokes a second FIS-update-contract request before the very first request (either a FIS-demand-contract or a FIS-update-contract request) has been confirmed, the FIS-demand-contract will be rejected.

2.4.7.4.1 The FIS-ground-user shall use the value received in the *FISContractNumber* parameter of the FIS-update-contract indication for the *FISContractNumber* parameter of any ground-initiated request or response related to that FIS update contract.

2.4.7.4.2 The same type of FIS information (i.e. “ATIS” for version 1 of the FIS-ASEs) shall be identified by the FIS-air-user and FIS-ground-user when invoking the FIS-demand-contract request and response primitives.

2.4.7.4.3 Upon receipt of a FIS-update-contract indication, if the FIS-ground-user is able to accept the contract in full then:

2.4.7.4.3.1 If the FIS-ground-user is not able to send the first FIS-report within the response time specified in 2.4.7.2.2., it shall:

a) invoke the FIS-update-contract response with the *Result* parameter set to the abstract value “positive acknowledgment”, and

b) invoke the FIS-report request later.

2.4.7.4.3.2 **Recommendation.**— *If the FIS-ground-user is not able to send the FIS report within the response time, it should invoke the FIS-update-contract response within the response time and then invoke the FIS-report request within 10 seconds.*

2.4.7.4.3.3 If the FIS-ground-user is able to send the FIS-report report within the response time specified in 2.4.7.2.2., it shall invoke the FIS-update-contract response with the following parameters:

a) the *Result* parameter set to the abstract value “accepted”, and

b) the FIS report in the *FIS Information* parameter.

2.4.7.4.3.4 If the FIS-ground-user does not support the update-contract function but the requested FIS report is available, it shall invoke the FIS-update-contract response with the following parameters:

a) the *Result* parameter set to the abstract value “rejected”,

b) the *RejectReason* parameter set to the abstract value "update contract function not supported by the FIS-ground-user", and

c) the FIS report in the *FISInformation* parameter.

2.4.7.4.4 Upon receipt of a FIS-update-contract indication, if the FIS-ground-user cannot comply with the update contract request, it shall reject the contract by invoking a FIS-update-contract response with the following parameters:

a) the abstract value "rejected" as *Result* parameter, and

b) one of the following abstract values "cannot comply", "FIS service not available", "error detected in the FIS request" or "undefined", as *RejectReason* parameter.

2.4.7.4.5 The FIS-ground-user shall invoke the FIS-report request each time the FIS information requested in the FIS-update-contract indication is updated until such time as the contract is canceled or aborted.

2.4.7.4.6 The FIS-air-user shall be allowed to reuse the value of a *FISContractNumber* used in a FIS-update-contract request for a new FIS contract:

a) upon receipt of a FIS-update-contract confirmation with a *Result* parameter containing the abstract value "rejected",

b) upon receipt of a FIS-user-abort indication or a FIS-provider-abort indication,

c) upon receipt of a FIS-cancel-update confirmation or a FIS-cancel-contracts confirmation,

d) upon receipt of a FIS-cancel-update indication, or

e) after invocation of a FIS-user-abort request.

2.4.7.5 Air And Ground Cancellation Of a Single FIS Update Contract

Note 1.— There is no provision for the cancellation of a single FIS demand contract.

Note 2.— Both FIS-air-user and FIS-ground-user are capable of initiating the FIS-cancel-update-contract service.

Note 3.— When invoking the FIS-cancel-update-contract request, if the FIS-air-user does not provide in the FISContractNumber parameter a value identifying an active FIS update contract, the request is rejected.

Note 4.— If the FIS-air-user invokes a FIS-cancel-update-contract request before the very first request (either a FIS-demand-contract or a FIS-update-contract-request) has been confirmed, the FIS-cancel-update-contract request will be rejected.

2.4.7.5.1 Upon receipt of a FIS-cancel-update-contract indication, the FIS-user shall cancel the FIS update contract identified by the *FISContractNumber* parameter.

2.4.7.6 Air cancellation of FIS contracts of the same type

Note 1.— Only the FIS-air-user is capable of initiating the FIS-cancel-contracts service.

Note 2.— If the FIS-air-user invokes a FIS-cancel-contracts request before the very first request (either a FIS-demand-contract or a FIS-update-contract-request) has been confirmed, the FIS-cancel-contracts request will be rejected.

2.4.7.6.1 When the FIS-ground-user receives a FIS-cancel-contracts indication, it shall cancel any FIS demand contract and FIS update contract for the type(s) identified in the indication (e.g. ATIS) in force with that aircraft.

2.4.7.7 FIS-user-abort

Note 1.— Both FIS-air-user and FIS-ground-user are capable of initiating the FIS-user-abort service.

Note 2.— If an active FIS-user requires to stop brutally any activity of the current FIS-ASE with the peer FIS-ASE with a possible loss of FIS messages currently in transit in the communication system, it invokes the FIS-user-abort request.

Note 3.— After the FIS-user-abort request is invoked, the FIS-user is no more considered by the FIS-AE as an active user.

2.4.7.8 Parameter Value, Unit, Range and Resolution

2.4.7.8.1 A FIS-user shall interpret variable unit, range and resolution as defined in 2.4.4.

2.4.8 SUBSETTING RULES

2.4.8.1 General

Note.— 2.4.8 specifies conformance requirements which all implementations of the FIS protocol obey.

2.4.8.1.1 An implementation of either the FIS ground based service or the FIS air based service claiming conformance to 2.4 shall support the FIS protocol features as shown in the tables below.

Note.— The 'status' column indicates the level of support required for conformance to the FIS-ASE protocol described in this part. The values are as follows:

a) ***'M'*** *mandatory support is required,*

b) ***'O'*** *optional support is permitted for conformance to the FIS protocol,*

c) ***'N/A'*** *the item is not applicable, and*

d) ***'C.n'*** *the item is conditional where n is the number which identifies the condition which is applicable.*

Table 2.4.8-1. FIS Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 2.4.8-2. FIS Protocol Options

	Status	**Associated Predicate**
FIS-air-ASE	C.1	FIS/air
FIS-ground-ASE	C.1	FIS/ground
FIS Update Contract Function supported by the FIS-ground-user	O	FIS-user/UC
FIS Update Contract supported by the FIS-ground-ASE	if (FIS/ground) M, else O	UC-FU

C.1: a conforming implementation will support one and only one of these two options.

Table 2.4.8-3/a. FIS-air-ASE conformant configurations

	List of Predicates	**Functionality Description**
I	FIS/air	a FIS-air-ASE supporting the FIS demand contract (core functions)
II	FIS/air + UC-FU	a FIS-air-ASE supporting core functions and the FIS update contract

Table 2.4.8-3/b. FIS-ground-ASE conformant configurations

	List of Predicates	**Functionality Description**
I	FIS/ground	a FIS-ground-ASE fully supporting the FIS demand contract and functionality for receiving and indicating that the FIS Update Contract is not supported
II	FIS/ground + FIS-user/UC	a FIS-ground-ASE fully supporting both types of FIS contract

Table 2.4.8-4. Supported FIS Service Primitives

	Sending (req,[cnf])	**Receiving (ind, [rsp])**
FIS-demand-contact	if (FIS/air) M, else N/A	if (FIS/ground) M, else N/A
FIS-update-contact	if (FIS/air and UC-FU) M, else N/A	if (FIS/ground) M, else N/A
FIS-report	if (FIS/ground) M, else N/A	if (FIS/air) M, else N/A
FIS-cancel-contracts	if (FIS/air) O, else N/A	if (FIS/ground) M else N/A
FIS-cancel-update-contract	if (FIS/air and UC-FU) or (FIS/ground and FIS-user/UC) M, else N/A	if (FIS/air and UC-FU) or (FIS/ground and FIS-user/UC) M, else N/A
FIS-user-abort	M	M
FIS-provider-abort	N/A	M

Table 2.4.8-5. Supported FIS APDUs

	Sender	**Receiver**
[FISDownlinkAPDU] fISRequest - demand contract	if (FIS/air) M, else N/A	if (FIS/ground) M, else N/A
[FISDownlinkAPDU] fISRequest - update contract	if (FIS/air and UC-FU) M, else N/A	if (FIS/ground) M, else N/A
[FISDownlinkAPDU] fISCancelUpdateContract	if (FIS/air and UC-FU) M, else N/A	if (FIS/ground) M, else N/A
[FISDownlinkAPDU] fISCancelUpdateAccept	if (FIS/air and UC-FU) M, else N/A	if (FIS/ground) M, else N/A
[FISDownlinkAPDU] fISCancelContracts	if (FIS/air) O, else N/A	if (FIS/ground) M, else N/A
[FISDownlinkAPDU] fISAbort	if (FIS/air) M, else N/A	if (FIS/ground) M, else N/A
[FISUplinkAPDU] fISAccept	if (FIS/ground) M, else N/A	if (FIS/air) M, else N/A
[FISUplinkAPDU] fISReject	if (FIS/ground) M, else N/A	if (FIS/air) M, else N/A
[FISUplinkAPDU] fISReport	if (FIS/ground) m else N/A	if (FIS/air) M, else N/A
[FISUplinkAPDU] fISCancelUpdate Contract	if (FIS/air and UC-FU) or (FIS/ground and FIS-user/UC) M, else N/A	if (FIS/air and UC-FU) or (FIS/ground and FIS-user/UC) M, else N/A
[FISUplinkAPDU] fISCancelUpdateAccept	if (FIS/air and UC-FU) or (FIS/ground and FIS-user/UC) M, else N/A	if (FIS/air and UC-FU) or (FIS/ground and FIS-user/UC) M, else N/A
[FISUplinkAPDU] fISCancelContractsAccept	if (FIS/ground) M, else N/A	if (FIS/air) M, else N/A
[FISUplinkAPDU] fISAbort	if (FIS/ground) M, else N/A	if (FIS/air) M, else N/A