Doc 9880-AN/466
PART I

MANUAL ON DETAILED TECHNICAL SPECIFICATIONS FOR THE AERONAUTICAL TELECOMMUNICATION NETWORK (ATN) using ISO/OSI standards and protocols

PART I – AIR-GROUND APPLICATIONS

1st edition *(unedited)*

NOTICE TO USERS

This document is an unedited advance version of an ICAO publication as approved, in principle, by the Secretary General, which is rendered available to the public for convenience. The final edited version may still undergo alterations in the process of editing. Consequently, ICAO accepts no responsibility or liability of any kind should the final text of this publication be at variance from that appearing here.

Advance edition *(unedited)*

Foreword

This manual replaces the *"Manual of technical provisions for the Aeronautical Telecommunication Network (ATN)",* Doc 9705 – third edition. Amendments to Doc 9705 are incorporated. These amendments were necessary as a result of ongoing validation, and operational experience gained during implementation of elements of the ATN. These amendments were reviewed at the ACP Working Group of the Whole #1 Meeting in June 2005 and further updated at the ACP Working Group N/06 Meeting held in July 2006. Relevant background material is available in the reports of these meetings, which can be accessed at www.icao.int/anb/panels/acp.

The different parts of this manual will be published as and when the relevant sub-volumes of Doc 9705 have been updated and completed.

This manual contains the detailed technical specifications for the ATN, based on relevant Standards and protocols established by the International Organization for Standardization (ISO) and the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) for Open Systems Interconnection (OSI). A separate manual, addressing detailed technical specifications for the ATN, based on Standards developed by the Internet Society (ISOC) for the Internet Protocol Suite (IPS) is in preparation, together with draft Standards and Recommended Practices (SARPs) for the ATN/IPS. Where necessary and to avoid duplication of essential material, the IPS manual will refer to this manual, as required.
This manual will be published in the following parts:

Part I	Air-ground applications (Doc 9705/sub-volume II)
Part II	Ground-ground applications (Doc 9705/sub-volume III)
Part III	Internet communication service, including upper layer communications service (Doc 9705/sub-volumes IV and V).
Part IV	Directory service, security services, systems management, Identifier registration and definitions (Doc 9705/sub-volumes I, VI, VII, VIII and IX).

With the publication of each part of this manual, the relevant sub-volumes of Doc 9705 will become obsolete.

— — — — — — — —

Chapter 1 – Introduction

1.1 Part I of this manual replaces and updates sub-volume II of the ICAO *Manual of Technical Provisions for the Aeronautical Telecommunication Network (ATN)* (Doc 9705). It includes amendments to the third edition to Doc 9705.

1.2 Included in this manual are provisions for:

Chapter 2 Context Management (CM)

Chapter 3 Controller-pilot data link communications (CPDLC)

Chapter 4 Flight Information Services (FIS) (in preparation)

Chapter 5 Automatic Dependent Surveillance (ADS-C) (in preparation)

Chapter 6 ATN Message Integrity Check Algorithm

1.3 **Dialogue Service (DS)**

1.3.1 Throughout this document references to dialogue services (D-START, D-DATA, D-END, D-ABORT, D-P-ABORT) are references to the Dialogue Service specified in Doc 9705, Sub-volume IV, Chapter 4.2.

1.4 **Context management (CM)**

1.4.1 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.

1.4.2 CM applications need to be considered in the context of the guidance material for the data link initiation capability (DLIC), as provided in the *"Manual of air traffic services data link applications"* (Doc 9694, Part II). The DLIC process supports addressing requirements for ATS such as CPDLC, FIS and ADS-C.

1.5 **Controller-pilot data link communications (CPDLC)**

1.5.1 The CPDLC application allows data link communications between controllers and pilots.

1.5.2 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. The CPDLC application supports a variety of operational data link services, depending upon the CPDLC message and operational procedures applied.

1.5.3 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.

1.5.4. The CPDLC application includes an end-to-end protection of message integrity by an application level integrity check that also provides assurance of correct delivery.

1.5.5 The CPDLC application complies with the provisions in the *Procedures for Air Navigation Services — Air Traffic Management* (PANS-ATM, Doc 4444), Chapter 14 (Controller-pilot data link communications (CPDLC)) and Appendix 5 (CPDLC message set), up to and including Amendment 4 to the 14th edition.

1.5.6 The CPDLC application also complies with the provisions in the *Manual of Air Traffic Services Data Link Applications* (Doc 9694), First Edition — 1999), Part IV.

1.5.7 The CPDLC application supports a variety of operational data link services, depending upon the message subset and operational procedures applied.

1.6 Flight information service

(to be developed)

1.7 ADS-Broadcast

(to be developed)

DEFINITIONS

Data link initiation capability (DLIC) A data link application that provides the ability to exchange addresses, names and version numbers necessary to initiate data link applications.

Note 1.— DLIC is an operational service which is being enabled over the ATN through CM.

Note 2.— For other definitions, see Doc 9705, Sub-volume I.

— — — — — — — —

Chapter 2 – Context management application

(See mapping table for conversion of current paragraph numbers of Doc 9705 – 3rd edition into paragraph numbers of Doc 9880)

2. CONTEXT MANAGEMENT APPLICATION

2.1 INTRODUCTION

2.1.1 General

2.1.1.1 This chapter contains the following paragraphs:

2.2 *GENERAL REQUIREMENTS*, which contains CM ASE Version Number and error processing requirements.

2.3 *ABSTRACT SERVICE DEFINITION, which* contains the description of the abstract service provided by the CM Application Service Element (CM-ASE).

2.4 *FORMAL DEFINITION OF MESSAGES*, which contains the formal definition of messages exchanged by CM-ASEs using Abstract Syntax Notation Number One (ASN.1).

2.5 *PROTOCOL DEFINITION, which* 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.

2.6 *COMMUNICATION REQUIREMENTS, which* contains the requirements that the CM application imposes on the underlying communication system.

2.7 *CM USER REQUIREMENTS, which* contains requirements imposed on the user of the CM-SE service.

2.8 *SUBSETTING RULES* which contains the conformance requirements which all implementations of the CM protocol obey.

2.1.2 Functional Description of the context management (CM) applications

2.1.2.1 Logon Functional Description

2.1.2.1.1 The logon function provides a means for an aircraft and ground system to exchange ATN application information. The logon function has two variants - secure and unsecure. Secure logons are only available to version 2 CM systems. Version 2 CM systems have the option of performing secure or unsecure logon functions.

2.1.2.1.2 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. Additionally, if the aircraft is a version 2 CM, it can provide the Security Requirements parameter. This parameter will have different values depending on whether or not the logon will be requesting secure or unsecure services. Version 1 CM

applications cannot provide the Security Requirements parameter. The user data in the CM-logon request is the same for both secure and unsecure services. 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. For version 2 CM, if a secure logon service is requested and is able to be supported, the response to the CM-logon request will also include additional security information as well as the Security Requirements parameter. However, a successful secure Logon function does not guarantee that other applications (such as ADS, CPDLC and FIS) will be able to be run in a secure mode; that will be dependent on local security policy.

2.1.2.1.3 For version 2 CM, an "emulated version" CM-logon can also be performed. This option may be used in order to start a CM-logon service with a known earlier version ground CM application.

2.1.2.1.4 Up to a maximum of 256 applications can be supported.

2.1.2.1.5 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.

2.1.2.1.6 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.

2.1.2.1.7 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. For version 2 CM, if security is supported, the CM Logon Response message also contains the key for each application as well as an indication of the key's domain applicability.

2.1.2.2 Server Facility Query Functional Description

2.1.2.2.1 This function is only available for version 2 CM, and may be secure or unsecure. The Server Facility Query function can only be air initiated. The aircraft can use the server query function to provide application information to a ground system supporting CM server functionality. In addition, the aircraft can specify up to eight facility designations with which it wishes to perform data link services. In response, the ground provides application information for each of the facility designations specified. If secure services are supported and requested, the ground will also reply with key information and domain usage information for each application. However, a successful secure Server Facility Query function does not guarantee that other applications (such as ADS, CPDLC and FIS) will be able to be run in a secure mode; that will be dependent on local security policy. The ground may also reply with a message saying the service is not supported or unavailable.

2.1.2.2.2 Up to a maximum of 256 applications can be supported.

2.1.2.2.3 Each time a server facility query is accomplished between a given aircraft and a ground system, the latest exchanged information replaces any previous information for each indicated application.

2.1.2.2.4 The CM Server Facility Query Request message provides required flight plan information, the aircraft's CM application name and address, up to eight facility designations with which the aircraft wishes to perform data link services, and information for each application on the aircraft 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.

2.1.2.2.5 The CM Server Facility Query Response message provides application information for each of the facility designations as indicated by the server query request. For each application that can be air initiated, the ground system must provide the application name, version number and address. For each application which is only ground initiated, the ground system must provide the application name and version number. Additionally, if secure services are supported and requested, the ground system must also provide key information and domain usage applicability for each application if available.

2.1.2.3 Server Facility Update Functional Description

2.1.2.3.1 This function is only available for version 2 CM, and may be secure or unsecure. This function provides a method for the ground system to update application information for up to eight facility designations. This function assumes that the server facility query function has been accomplished.

2.1.2.3.2 The CM Server Facility Update message can provide updated ground information for up to 256 applications for each of up to eight different facilities. For each updated application, the ground provides the application's name, version number and address, if applicable. Additionally, if secure services are supported and requested, the ground system must also provide key information and domain usage applicability for each application if available.

2.1.2.4 Update Functional Description

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

2.1.2.4.2 For version 2 CM, both secure and unsecure Update functions are available.

2.1.2.4.3 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. For version 2 secure updates, security information for each application (key and domain usage indication) is also included. However, a successful secure Update function does not guarantee that other applications (such as ADS, CPDLC and FIS) will be able to be run in a secure mode; that will be dependent on local security policy.

2.1.2.5 Contact Functional Description

2.1.2.5.1 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. For version 2 CM, the contact function may be secure or unsecure. There are no differences in the user data provided by the CM-ground-user or CM-air-user; only the provision of the Security Requirements parameter is different.

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

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

2.1.2.6 Forwarding Functional Description

2.1.2.6.1 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.

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

2.1.2.6.3 For version 2 CM, the CM forward function may be secure or unsecure. In addition, version 2 CM forward functions includes the aircraft's CM version number along with the aircraft's CM Logon information.

2.1.2.7 Registration Functional Description

2.1.2.7.1 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.

2.1.2.7.2 For version 2 CM, registration also includes additional security functionality. For CM ground systems, this includes retrieving the key agreement public keys and domain usage information for each supported application from the directory maintaining that information, and making it available to the user of each application (such as CPDLC) as well as to the secure Dialogue Service. For CM aircraft systems, this includes making the key agreement public keys and domain information received from a ground system available to the user of each application (such as CPDLC) as well as to the secure Dialogue Service Provider. These steps are necessary in order for other secure services to be used. Additionally, local procedures may call for further directory interaction in order to retrieve information from or submit updated information to the directory.

2.1.2.7.3 There are no standard message exchanges for this function.

2.2 GENERAL REQUIREMENTS

2.2.1 CM ASE Version Number

The CM-air-ASE and CM-ground-ASE version numbers shall both be set to one (basic CM functionality) or two (extended CM functionality).

Note.— The extended CM functionality provides additional capability to CM-users: it allows a CM-air-user to request information for multiple ground facilities, a CM-ground-user to give an aircraft

information for multiple ground facilities, authentication and data integrity services to be performed for air-ground and ground-ground dialogues, and the requirement to obtain information from a directory.

2.2.2 Error Processing Requirements

2.2.2.1 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.

2.2.2.2 In the event of a CM-user invoking a CM service primitive when the CM-ASE is not in a state specified in 2.5, the CM-user shall be notified.

2.3 THE ABSTRACT SERVICE

2.3.1 Service Description

2.3.1.1 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.— 2.3 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.— 2.3 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 2.7.

Note 3.— Figure 2.3-1 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.

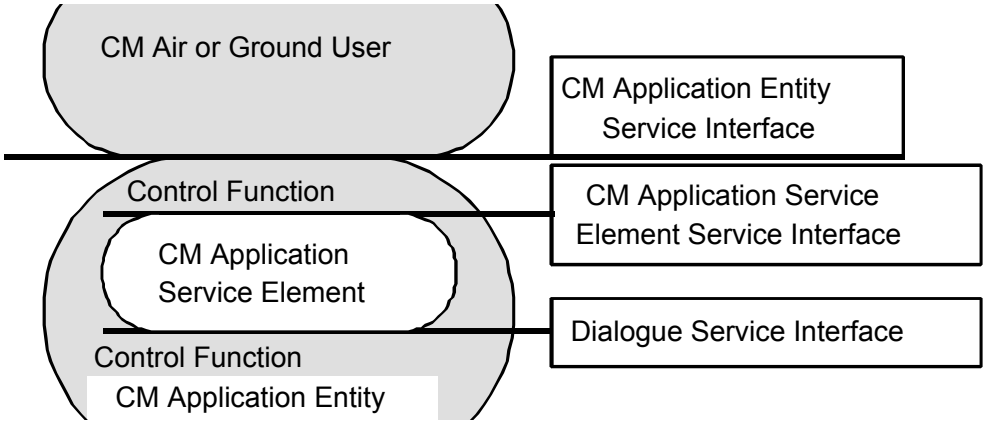


Figure 2.3-1. Functional Model of the CM Application

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 Doc 9705, Sub-volume IV, paragraph 4.2.

2.3.2 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.

2.3.2.1 The CM-ASE abstract service shall consist of a set of the following services as allowed by the sub-setting rules in 2.8:

a) CM-logon service as defined in 2.3.3;

b) for CM version 2, CM-server-facility-query as defined in 2.3.4;

c) CM-update service as defined in 2.3.5;

d) for CM version 2, CM-server-facility-update as defined in 2.3.6;

e) CM-contact service as defined in 2.3.7;

f) CM-end service as defined in 2.3.8;

g) CM-forward service as defined in 2.3.9;

h) CM-user-abort service as defined in 2.3.10; and

i) CM-provider-abort service as defined in 2.3.11.

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:

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 2.4 for this field.

2.3.3 CM-logon Service

Note 1.— 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.

Note 2.— For CM version 2 CM-logon service, the CM-air-user will set the Security Requirements *parameter as required for the particular airspace. A version 2 CM-air-user may attempt either a secure or unsecure CM-logon service. Additionally, if it is known that the peer CM ground system is an earlier version CM, then the sending CM-air-user may indicate to the CM-air-ASE that it wishes to run in a previous version mode through the* Emulated Version *parameter. If this parameter is used, no other version 2 CM services may be invoked with that ground system.*

2.3.3.1 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 2.3-1.

Table 2.3-1. 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(=)
Security Required	M	M(=)	C(=)	
Emulated Version	U			

2.3.3.2 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 2.3-2.

Table 2.3-2. 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	
Security Required	M	
Emulated Version	U	

2.3.3.3 Facility Designation

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

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

2.3.3.4 Aircraft Address

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

2.3.3.4.1 The *Aircraft Address* parameter value shall conform to the abstract syntax of the ICAO 24 bit aircraft address.

2.3.3.5 **CM ASE Version Number**

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

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

2.3.3.5.2 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.

2.3.3.5.3 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.— 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.

2.3.3.6 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.

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

2.3.3.7 Logon Response

Note.— This parameter contains information for each requested data link application for which the ground is able to provide data link service. For the CM version 2 secure CM-logon service, this parameter will also contain key and domain usage information for each application.

2.3.3.7.1 For CM version 1, the *Logon Response* parameter value shall conform to the ASN.1 abstract syntax CMLogonResponse.

2.3.3.7.2 For CM version 2, the *Logon Response* parameter value shall conform to the ASN.1 abstract syntax CMSecureLogonResponse.

Note.— For unsecure CM version 2, no security information is provided in the CMSecureLogonResponse.

2.3.3.8 Class of Communication Service

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

2.3.3.8.1 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.

2.3.3.9 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.

2.3.3.9.1 If provided by the CM-ground-user this parameter shall have the abstract value "maintain".

2.3.3.10 Security Required

Note 1.— The Security Required *parameter contains an indication of whether or not a CM version 2 requires security. It is used by the requester of the version 2 CM-logon service and is indicated to the responder. It is not used by version 1 CM.*

Note 2.— If the received Security Required parameter is not as expected per the local security policy, the CM-ground-ASE will abort. The CM-ground-user will use the indicated Security Required parameter to determine whether or not to return key information to the requesting aircraft. This is necessary for the case where a ground system can provide both secure and unsecure services.

2.3.3.10.1 The value of the *Security Required* parameter provided by the CM-air-user shall be indicated to the CM-ground-user.

2.3.3.10.2 When the CM-air-user requires a secure CM-logon service, the *Security Required* parameter shall have the abstract value "Secured Dialogue supporting key management".

Note.— This is to give an indication to the CM-ground-user that security is to be used.

2.3.3.10.3 When the CM-air-user requires an unsecure CM-logon service or the CM-air-user is communicating with a CM version 1 ground system, the Security Required parameter shall have the abstract value "No security".

Note.— This is to give an indication to the CM-ground-user that security is not to be used. However, the CM-ground-user will have the ultimate decision in whether or not security is required. If security is required by the ground system but not requested by the aircraft, the aircraft may be denied data link services.

2.3.3.11 Emulated Version

Note 1.— This parameter is used by a CM-air-user to indicate that the CM-air-ASE is to operate in a degraded mode, i.e. to emulate a previous CM version. If this parameter is provided, the CM-air-user must ensure that no services incompatible with the CM version emulated are invoked. This parameter is not used by version 1 CM.

Note 2.— This parameter is not normally provided for a first logon unless there is explicit a priori knowledge of the CM-ground-ASE version number.

Note 3.— The value supplied by the CM-air-user must be less than the actual CM-air-ASE version number.

2.3.3.11.1 If provided by the CM-air-user, this parameter shall conform to an abstract integer value from 1 to 255.

2.3.4 CM-server-facility-query Service

Note.— The CM-server-facility-query service allows version 2 CM-air-users to initiate either secure or unsecure data link services with a CM server. This service is similar to a CM-logon service. However, unlike the CM-logon service, the CM-server-facility-query service does not perform version negotiation. In addition, the CM-server-facility-query service can be invoked either with or without a dialogue in place. The CM-air-user provides information on each data link application for which it desires a data link service, as well as the facility designations of the end systems with which it wants data link service. The CM-ground-user responds indicating whether or not the CM-server-facility-query was successful, and if successful, includes information of each data link application supported by the requested facilities. It is a confirmed service. This service is not used for version 1 CM.

2.3.4.1 The CM-server-facility-query service shall contain the primitives and parameters as presented in Table 2.3-2a.

Table 2.3-2a. CM-server-facility-query Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Facility Designation	C			
Aircraft Address	C	C(=)		
Server Facility Query Request	M	M(=)		
Server Facility Query Response			M	M(=)
Class of Commu nication Service	U			
Maintain Dialogue			U	C(=)
Security Required	C	C(=)		

2.3.4.2 Facility Designation

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

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

2.3.4.2.2 If a CM dialogue does not exist when a CM-air-user invokes the CM-server-facility-query request, the CM-air-user shall provide the *Facility Designation* parameter value.

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

2.3.4.3 Aircraft Address

Note.— This parameter contains the ICAO 24 bit aircraft address of the aircraft initiating the CM-server-facility-query service.

2.3.4.3.1 The *Aircraft Address* parameter value shall conform to the abstract syntax of the ICAO 24 bit aircraft address.

2.3.4.3.2 If a CM dialogue does not exist when a CM-air-user invokes the CM-server-facility-query request, the CM-air-user shall provide the *Aircraft Address* parameter value.

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

2.3.4.4 Server Facility Query Request

Note.— The Server Facility Query 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;

b) the facility designations of up to eight facilities with which the aircraft would like to perform data link services with; and

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

The *Server Facility Query Request* parameter value shall conform to the ASN.1 abstract syntax CMServerFacilityQueryRequest.

2.3.4.5 Server Facility Query Response

Note.— This parameter contains application information for each of the facilities as requested in the Server Facility Query Request *parameter. If application information for one of the facilities is not available, then no application information is returned for that facility. If there is no access to a directory, then the reason for the lack of access will be returned (i.e. either a directory service is not supported or the service is temporarily unavailable). Additionally, if a collision occurs and preference is given to the ground system's*

CM service, then that condition will be indicated to the CM-air-user. The Server Facility Query Response *parameter will contain key information only if security is supported.*

2.3.4.5.1 The *Server Facility Query Response* parameter value shall conform to the ASN.1 abstract syntax CMServerFacilityQueryResponse.

2.3.4.6 Class of Communication Service

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

2.3.4.6.1 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.

2.3.4.7 Maintain Dialogue

Note 1.— This parameter is used to indicate whether or not the requested CM dialogue is to remain open after a Server Facility Query 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 and a dialogue is not already in place.

Note 4.— This parameter is not used if a dialogue is already in place.

2.3.4.7.1 If provided by the CM-ground-user this parameter shall have the abstract value "maintain".

2.3.4.8 Security Required

Note 1.— The Security Required parameter contains an indication of whether or not a CM version 2 requires security. It is used by the requester of the CM-server-facility-query service and is indicated to the responder.

Note 2.— If the received Security Required parameter is not as expected in the indication primitive, the CM-ground-ASE will abort.

2.3.4.8.1 If a CM dialogue does not exist when a CM-air-user invokes the CM-server-facility-query request, the CM-air-user shall provide the *Security Required* parameter.

Note.— The CM-server-facility-query service does not use this parameter if a dialogue exists.

2.3.4.8.2 When the *Security Required* parameter is provided by the CM-air-user, the same value shall be indicated to the CM-ground-user.

2.3.4.8.3 When the CM-air-user requires a secure CM-server-facility-query service and no previous session key has been established with the CM ground system which is to be contacted, the *Security Required*

parameter shall have the abstract value "Secured Dialogue supporting key management".

Note.— A session key may be established with a secure CM-logon, secure CM-update, secure CM-server-facility-update, or secure CM-server-facility-query. If one of these secure services has not already been successfully performed with the ground system, then a session key needs to be established, and the "Secured Dialogue supporting key management" will tell the upper layers to establish a new session key. The CM-air-user does not establish the session key itself; that is handled as described in Doc 9705, Sub-volume VIII.

2.3.4.8.4 When the CM-air-user requires a secure CM-server-facility-query service and a previous session key has been established with the CM ground system which is to be contacted, the *Security Required* parameter shall have the abstract value "Secured Dialogue".

Note.— A session key may be established with a secure CM-logon, secure CM-update, secure CM-server-facility-update, or secure CM-server-facility-query. If one of these secure services has already been successfully performed with the ground system, then a session key is considered to be established, and the "Secured Dialogue" will tell the upper layers to continue using security with the established session key. The CM-air-user does not establish the session key itself; that is handled as described in Doc 9705, Sub-volume VIII.

2.3.4.8.5 When the CM-air-user requires a unsecure CM-server-facility-query service, the *Security Required* parameter shall have the abstract value "No security".

Note.— This is to give an indication to the CM-ground-user that security is not to be used. However, the CM-ground-user will have the ultimate decision in whether or not security is required. If security is required by the ground system but not requested by the aircraft, the aircraft may be denied data link services.

2.3.5 CM-update Service

Note 1.— 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.

Note 2.— For CM version 2 CM-update service, the CM-ground-user will set the Security Required parameter as appropriate in order to perform secure or unsecure services. This parameter is not used by version 1 CM.

2.3.5.1 The CM-update service shall contain the primitives and parameters as presented Table 2.3-3.

Table 2.3-3. CM-update Service Parameters

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

2.3.5.2 Aircraft Address

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

2.3.5.2.1 The *Aircraft Address* parameter value shall conform to the abstract syntax of the ICAO 24 bit aircraft address.

2.3.5.2.2 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.

2.3.5.3 Facility Designation

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

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

2.3.5.3.2 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.

2.3.5.4 Update Information

Note.— This parameter contains information on each updated data link application. For version 2 secure CM-update service, this parameter also contains key and domain usage indications for each application.

2.3.5.4.1 For CM version 1 or CM version 2 with no security required, the *Update Information* parameter value shall conform to the ASN.1 abstract syntax CMUpdate.

2.3.5.4.2 For CM version 2 with security required, the *Update Information* parameter value shall conform to the ASN.1 abstract syntax CMSecureUpdate.

2.3.5.5 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.

2.3.5.5.1 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.

2.3.5.6 Security Required

Note.— This parameter contains an indication of whether or not secure CM-update services will be used by a version 2 CM. This parameter is not used for CM version 1.

2.3.5.6.1 When the CM-ground-user requires a secure CM-update service and up-to-date security information has not already been established between that airborne system and the CM ground system (i.e. a current session key is not shared), the *Security Required* parameter shall have the abstract value “Secured Dialogue supporting key management”.

Note.— This is to give an indication to the CM-air-ASE that security is to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.5.6.2 When the CM-ground-user requires a secure CM-update service and up-to-date security information has already been established between that airborne system and the CM ground system (i.e. a current session key is shared), the Security Required parameter shall have the abstract value “Secured Dialogue”.

Note.— This is to give an indication to the CM-air-ASE that security is to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.5.6.3 When the CM-ground-user requires an unsecure CM-update service and the CM aircraft system is not Version 1, the *Security Required* parameter shall have the abstract value “No security”.

Note.— This is to give an indication to the CM-air-ASE that security is not to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.5.6.4 If a CM dialogue does not already exist between the CM ground system and CM aircraft system, and the CM aircraft system is not Version 1, then when a CM-ground-user invokes the CM-update request, the CM-ground-user shall provide the *Security Required* parameter.

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

2.3.5.6.5 When communicating with a version 1 CM aircraft, the CM-ground-user shall be prohibited from providing the *Security Required* parameter.

2.3.6 CM-server-facility-update Service

Note.— The CM-server-facility-update service allows version 2 CM servers to transmit updated application information for up to eight specific facilities to update previously co-ordinated CM-server-facility-query information. It is an unconfirmed service, and may be either secure or unsecure. This service is not used by version 1 CM.

2.3.6.1 The CM-server-facility-update service shall contain the primitives and parameters as presented in Table 2.3-3a.

Table 2.3-3a. CM-server-facility-update Service Parameters

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

2.3.6.2 Aircraft Address

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

2.3.6.2.1 The *Aircraft Address* parameter value shall conform to the abstract syntax of the ICAO 24 bit aircraft address.

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

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

2.3.6.3 Facility Designation

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

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

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

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

2.3.6.4 Server Facility Update Information

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

2.3.6.4.1 The *Server Facility Update Information* parameter value shall conform to the ASN.1 abstract syntax CMServerFacilityUpdate.

Note.— The Server Facility Update Information *parameter will contain key information only if security is supported.*

2.3.6.5 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-server-facility-update service does not use this parameter when a CM dialogue exists.

2.3.6.5.1 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.

2.3.6.6 Security Required

Note.— This parameter contains an indication of whether or not secure CM-server-facility-update services will be used.

2.3.6.6.1 When the CM-ground-user requires a secure CM-server-facility-update service and a previous CM-logon has not been performed with the CM ground system which is to be contacted, the *Security Required* parameter shall have the abstract value "Secured Dialogue supporting key management".

Note.— This is to give an indication to the CM-air-ASE that security is to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.6.6.2 When the CM-ground-user requires a secure CM-server-facility-update service and a previous CM-logon has been performed with the CM ground system which is to be contacted, the *Security Required* parameter shall have the abstract value "Secured Dialogue".

Note.— This is to give an indication to the CM-air-ASE that security is to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.6.6.3 When the CM-ground-user requires an unsecure CM-server-facility-update service, the *Security Required* parameter shall have the abstract value "No security".

Note.— This is to give an indication to the CM-air-ASE that security is not to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.6.6.4 If a CM dialogue does not exist when a CM-ground-user invokes the CM-server-facility-update request, the CM-ground-user shall provide the *Security Required* parameter.

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

2.3.7 CM-contact Service

Note 1.— 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.

Note 2.— For CM version 2 CM-contact service, the CM-ground-user will set the Security Required *parameter as appropriate in order to perform secure or unsecure services. This parameter is not used by version 1 CM.*

2.3.7.1 The CM-contact service shall contain the primitives and parameters as presented in Table 2.3-4.

Table 2.3-4. 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			
Security Required	C			

2.3.7.2 Aircraft Address

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

2.3.7.2.1 The *Aircraft Address* parameter value shall conform to the abstract syntax of the ICAO 24 bit aircraft address.

2.3.7.2.2 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.

2.3.7.3 Facility Designation

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

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

2.3.7.3.2 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.

2.3.7.4 Contact Request

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

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

CMContactRequest.

2.3.7.5 Contact Response

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

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

2.3.7.6 Class of Communication Service

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

2.3.7.6.1 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.

2.3.7.7 Security Required

Note.— The Security Required parameter contains an indication of whether or not a CM-ground-user requires security. It is used by the requester of the version 2 CM-contact service. It is not used by version 1 CM.

2.3.7.7.1 When the CM-ground-user requires a secure CM-contact service and a previous secure CM-logon has been performed with the CM ground system, the *Security Required* parameter shall have the abstract value "Secured Dialogue".

Note.— This is to give an indication to the CM-air-ASE that security is to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.7.7.2 When the CM-ground-user requires an unsecure CM-contact service or if the CM-ground-user is communicating with a version 1 CM aircraft, the *Security Required* parameter shall have the abstract value "No security".

Note.— This is to give an indication to the CM-air-ASE that security is not to be used. The CM-air-ASE can then make security decisions based on local security policy.

2.3.7.7.3 If a CM dialogue does not exist when a CM-ground-user invokes the CM-contact request, the CM-ground-user shall provide the *Security Required* parameter.

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

2.3.8 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.

2.3.8.1 The CM-end service shall contain the primitives as presented Table 2.3-5.

Table 2.3-5. CM-end Service Parameters

Parameter Name	Req	Ind
none		

2.3.9 CM-forward Service

Note 1.— 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.

Note 2.— For CM version 2, the CM-forward service can be secure or unsecure. This is indicated by the CM-ground-user setting the Security Required *parameter. If it is known that the peer CM ground system is an earlier version CM, then the sending CM-ground-user may indicate to the CM-ground-ASE that it wishes to run in a previous version mode through the* Emulated Version *parameter.*

2.3.9.1 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 2.3-6.

Table 2.3-6. 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
Security Required	M		
Emulated Version	U		

2.3.9.2 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 2.3-7.

Table 2.3-7. 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
Security Required	M	
Emulated Version	U	

2.3.9.3 Called Facility Designation

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

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

2.3.9.4 Calling Facility Designation

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

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

2.3.9.5 CM ASE Version Number

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

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

2.3.9.5.2 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.

2.3.9.5.3 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.— 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.*

2.3.9.6 Forward Request

Note.— This parameter contains information as provided in the CM Logon Request. For the CM version 2 CM-forward service, this parameter will also contain the aircraft's CM application version number.

2.3.9.6.1 For CM version 1 or for CM version 2 acting as a CM version 1 in emulation, the CM-forward service *Forward Request* parameter value shall conform to the ASN.1 abstract syntax CMForwardRequest.

2.3.9.6.2 For CM version 2 CM-forward service, the *Forward Request* parameter value shall conform to the ASN.1 abstract syntax CMEnhancedForwardRequest.

2.3.9.7 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.

2.3.9.7.1 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.

2.3.9.8 Result

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

2.3.9.8.1 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".*

2.3.9.9 Security Required

Note.— The Security Required *parameter contains an indication of whether or not a CM-ground-user requires security. It is used by the sending CM-ground-user of the version 2 CM-forward service. It is not used by version 1 CM, or by a version 2 CM emulating a version 1 CM.*

2.3.9.9.1 When the sending CM-ground-user requires a secure CM-forward service, the Security Required parameter shall have the abstract value "Secured Dialogue".

Note.— This is to give an indication to the peer CM-ground-ASE that security is to be used. The peer CM-ground-ASE can then make security decisions based on local security policy.

2.3.9.9.2 When the sending CM-ground-user requires an unsecure CM-forward service or if communicating with a version 1 CM ground system, the Security Required parameter shall have the abstract value "No security".

Note.— This is to give an indication to the peer CM-ground-ASE that security is not to be used. The peer CM-ground-ASE can then make security decisions based on local security policy.

2.3.9.10 Emulated Version

Note 1.— This parameter is used by a CM-ground-user to indicate that the sending CM-ground-ASE is to operate in a degraded mode, i.e. to emulate a previous CM version. This parameter is not used by version 1 CM.

Note 2.— This parameter is not normally provided for a first forward unless there is explicit a priori knowledge of the receiving ground CM application version number.

Note 3.— The value supplied by the sending CM-ground-user must be less than the actual sending CM-ground-ASE version number.

2.3.9.10.1 If provided by the sending CM-ground-user, this parameter shall conform to an abstract integer value from 1 to 255.

2.3.10 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.

2.3.10.1 The CM-user-abort service shall contain the primitives as presented Table 2.3-8.

Table 2.3-8. CM-user-abort Service Parameters

Parameter Name	Req	Ind
none		

2.3.11 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.

2.3.11.1 The CM-provider-abort service shall contain the primitives and parameters as presented Table 2.3-9.

Table 2.3-9. CM-provider-abort Service Parameters

Parameter Name	Ind
Reason	M

2.3.11.2 Reason

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

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

2.4 FORMAL DEFINITIONS OF MESSAGES

2.4.1 Encoding/decoding Rules

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

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

2.4.2 CM ASN.1 Abstract Syntax

2.4.2.1 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.

Note.— The ASN.1 module CMMessageSetVersion1 is used by both CM version 1 and CM version 2. The change in the ASN.1 for CM version 2 is the addition of PDUs for the CM-server-facility-query and CM-server-facility-update services and security enhancements, along with the supporting new service and security elements.

```
CMMessageSetVersion1 DEFINITIONS ::=

BEGIN

-- This is an import of the encoding of the subject's public key from Doc 9705, Sub-volume VIII for CM
Version 2

IMPORTS

        atnPKI-explicit FROM ATNObjectIdentifiers { iso(1) identified-organization(3)

                icao(27) atn(0) objectIdentifiers(0) } -- Defined in Doc 9705, Sub-volume IX,
paragraph 9.2.1.3

        ECPoint FROM ATN-PKI-Explicit  atnPKI-explicit      -- Defined in Doc 9705, Sub-volume
VIII

;  -- end of IMPORTS

-- ------------------------------------------------------------------------------------

-- CM Message Structure

-- ------------------------------------------------------------------------------------

--  Aircraft-generated messages

CMAircraftMessage ::=CHOICE

        {

        cmLogonRequest          [0]     CMLogonRequest,

        cmContactResponse       [1]     CMContactResponse,
```

```
        cmAbortReason                   [2]     CMAbortReason,
        ...,  -- the PDU after extensibility is CM Version 2 only
        cmServerFacilityQueryRequest  [3]     CMServerFacilityQueryRequest
        }

-- Ground-generated messages
CMGroundMessage ::=CHOICE
        {
        cmLogonResponse                 [0]     CMLogonResponse,
        cmUpdate                        [1]     CMUpdate,
        cmContactRequest                [2]     CMContactRequest,
        cmForwardRequest                [3]     CMForwardRequest,
        cmAbortReason                   [4]     CMAbortReason,
        cmForwardResponse               [5]     CMForwardResponse,
        ...,   -- All PDUs after extensibility are CM Version 2 only
        cmServerFacilityQueryResponse           [6]     CMServerFacilityQueryResponse,
        cmServerFacilityUpdate[7]       CMServerFacilityUpdate,
        cmSecureLogonResponse           [8]     CMSecureLogonResponse,
        cmSecureUpdate                  [9]     CMSecureUpdate,
        cmEnhancedForwardRequest        [10]    CMEnhancedForwardRequest
        }
-- ------------------------------------------------------------------------------
-- 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 Doc 9705, Sub-volume IX

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),
        protocol-error                       (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

CMEnhancedForwardRequest ::= SEQUENCE    -- CM Version 2 only
        {
        cmVersionNumber         VersionNumber,
        cmForwardRequest        CMLogonRequest,
        ...
        }

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
```

```
}

CMSecureLogonResponse ::= SEQUENCE  -- CM Version 2 only
{
facilityDesignation                 [0] FacilityDesignation          OPTIONAL,
secureAirInitiatedApplications      [1] SEQUENCE SIZE (1..256) OF SecAir OPTIONAL,
secureGroundOnlyInitiatedApplications                    [2]    SEQUENCE SIZE (1..256) OF
                                                                SecGnd OPTIONAL,
...
}

CMSecureUpdate ::= CMSecureLogonResponse  -- CM Version 2 only

CMServerFacilityQueryRequest ::= SEQUENCE   -- CM Version 2 only
{
aircraftFlightIdentification        [0] AircraftFlightIdentification,
cMLongTSAP                          [1] LongTsap,
groundInitiatedApplications         [2] SEQUENCE SIZE (1..256) OF  AEQualifierVersionAddress
                                                                                  OPTIONAL,
airOnlyInitiatedApplications        [3] SEQUENCE SIZE (1..256) OF  AEQualifierVersion
                                                                                  OPTIONAL,
requestedFacilities                 [4] SEQUENCE SIZE (1..8) OF FacilityDesignation,
airportDeparture                    [5] Airport                                   OPTIONAL,
airportDestination                  [6] Airport                                   OPTIONAL,
dateTimeDepartureETD                [7] DateTime                                  OPTIONAL,
...
}
```

CMServerFacilityQueryResponse ::= CHOICE -- CM Version 2 only

```
{
infoUnavailable     [0]     InfoUnavailable,
requestedInfo       [1]     SEQUENCE SIZE (1..8) OF RequestedInfo,
...
}
```

CMServerFacilityUpdate ::= SEQUENCE SIZE (1..8) OF RequestedInfo -- CM Version 2 only

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

```
DomainFlag ::= ENUMERATED   -- CM Version 2 only
        {
        keySharedInADM        (0),
        keyNotSharedInADM     (1),
        ...
        }

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

InfoUnavailable ::= ENUMERATED  -- CM Version 2 only
        {
        serverNotSupported            (0),
        serverUnavailable             (1),
        serviceInterrupted            (2),
        ...
        }

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

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

RequestedInfo ::= SEQUENCE  -- CM Version 2 only
        {
```

```
        facilityDesignation                 [0] FacilityDesignation,
        cMLongTSAP                          [1] LongTsap            OPTIONAL,
        airInitiatedApplications            [2] SEQUENCE SIZE (1..256) OF SecAir      OPTIONAL,
        groundOnlyInitiatedApplications     [3] SEQUENCE SIZE (1..256) OF SecGnd      OPTIONAL,
        ...
        }
Response ::= ENUMERATED
        {
        contactSuccess                      (0),
        contactNotSuccessful                (1)
        }

SecAir ::= SEQUENCE   -- CM Version 2 only
        {
        applicationInformation              [0] AEQualifierVersionAddress,
        keyAgreementPublicKey               [1] ECPoint                         OPTIONAL,
        domainFlag                          [2] DomainFlag                      OPTIONAL,
        ...
        }

SecGnd ::= SEQUENCE  -- CM Version 2 only
        {
        applicationInformation          [0]     AEQualifierVersion,
        keyAgreementPublicKey           [1]     ECPoint                     OPTIONAL,
        domainFlag                      [2]     DomainFlag                  OPTIONAL,
        ...
        }
```

```
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
```

2.5 PROTOCOL DEFINITION

2.5.1 Sequence Rules

2.5.1.1 With the exception of abort primitives, only the sequence of primitives described in Figures 2.5-1 through 2.5-26 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.

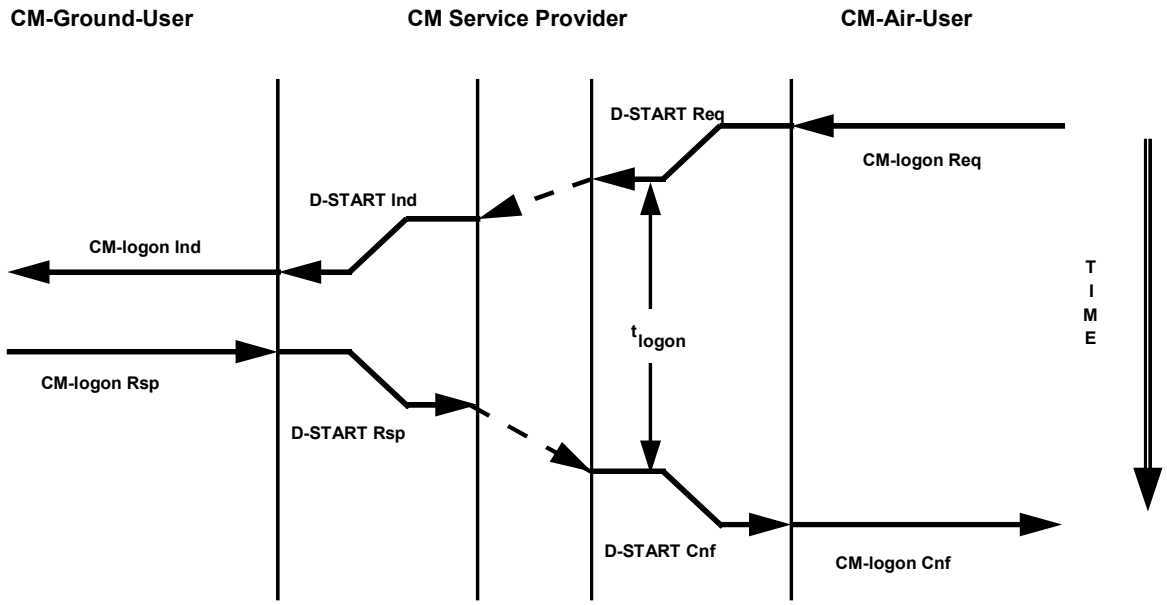


Figure 2.5-1. Sequence Diagram for CM-logon Service
CM-Air-ASE Version ≤ CM-Ground-ASE Version

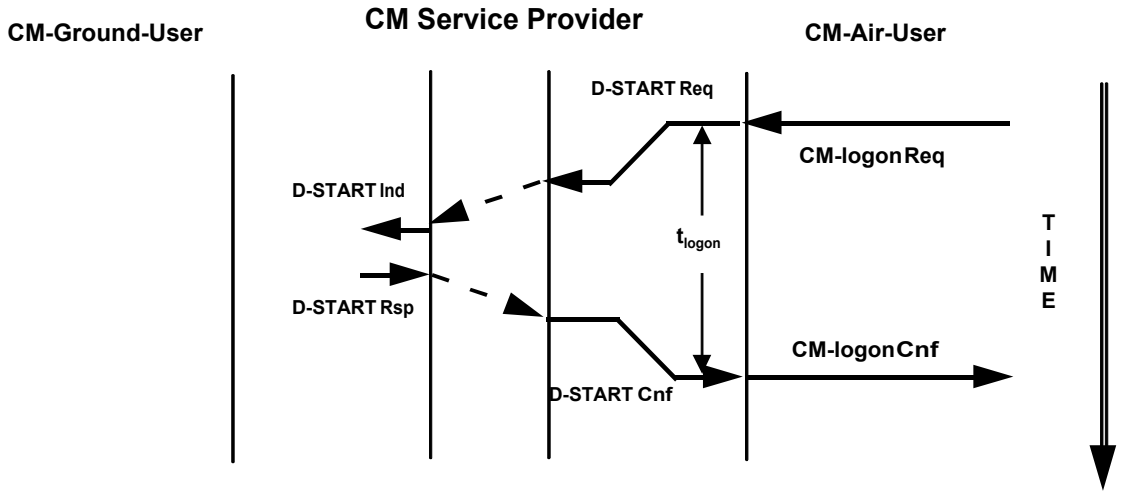


Figure 2.5-2. Sequence Diagram for CM-logon Service

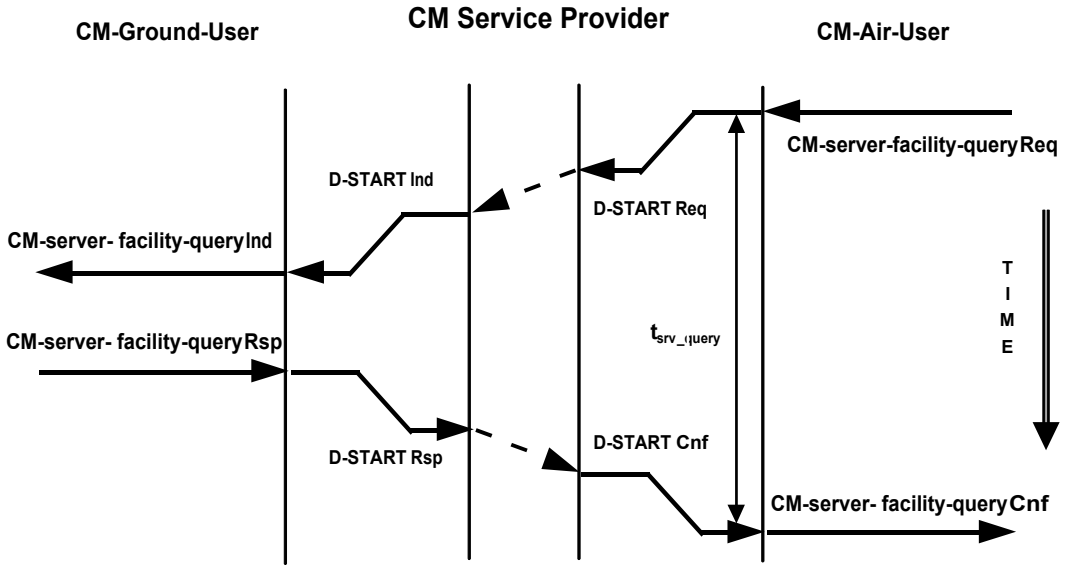


Figure 2.5-3a. CM-server-facility-query Service

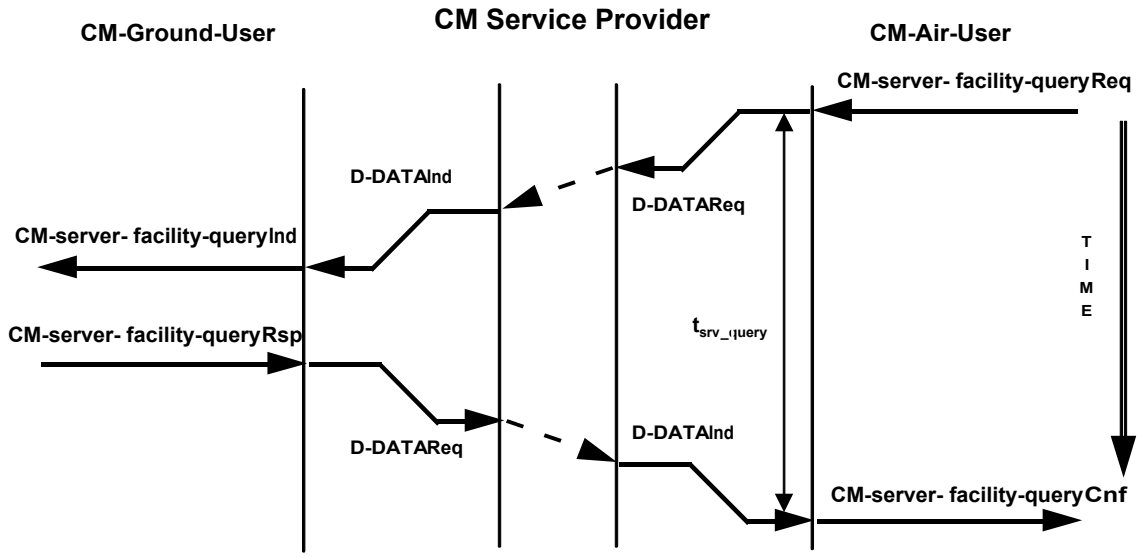


Figure 2.5-3b. CM-server-facility-query Service
Existing CM Dialogue (CM Version 2)

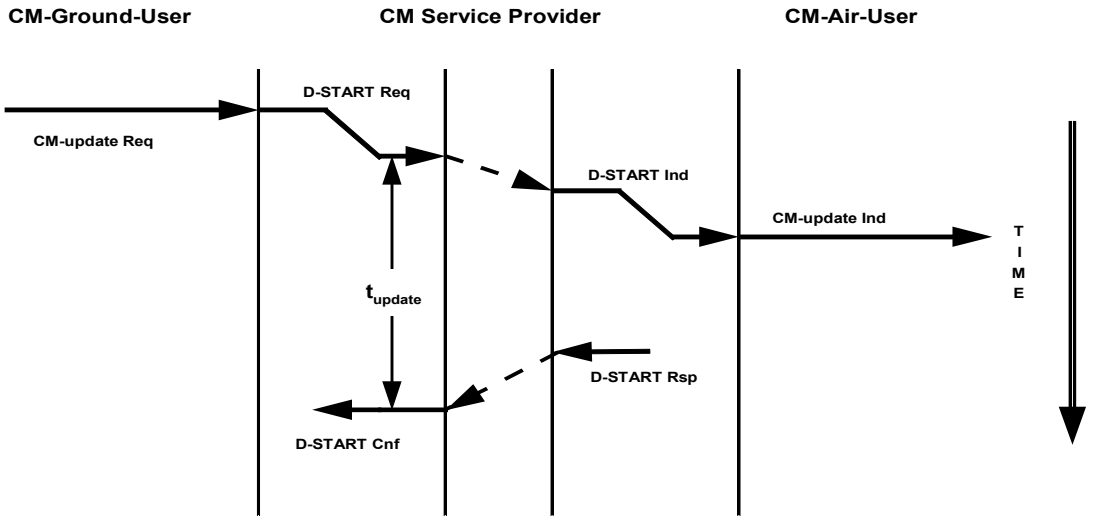


Figure 2.5-3c. Sequence Diagram for CM-update Service
No Existing CM Dialogue

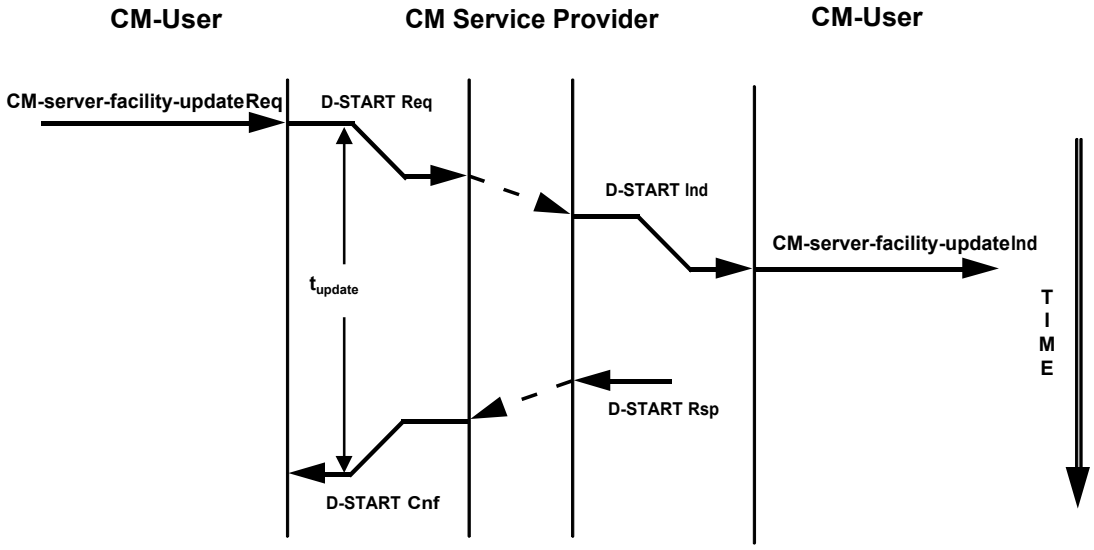


Figure 2.5-3d. Sequence Diagram for CM-server-facility-update Service No Existing CM Dialogue (CM Version 2)

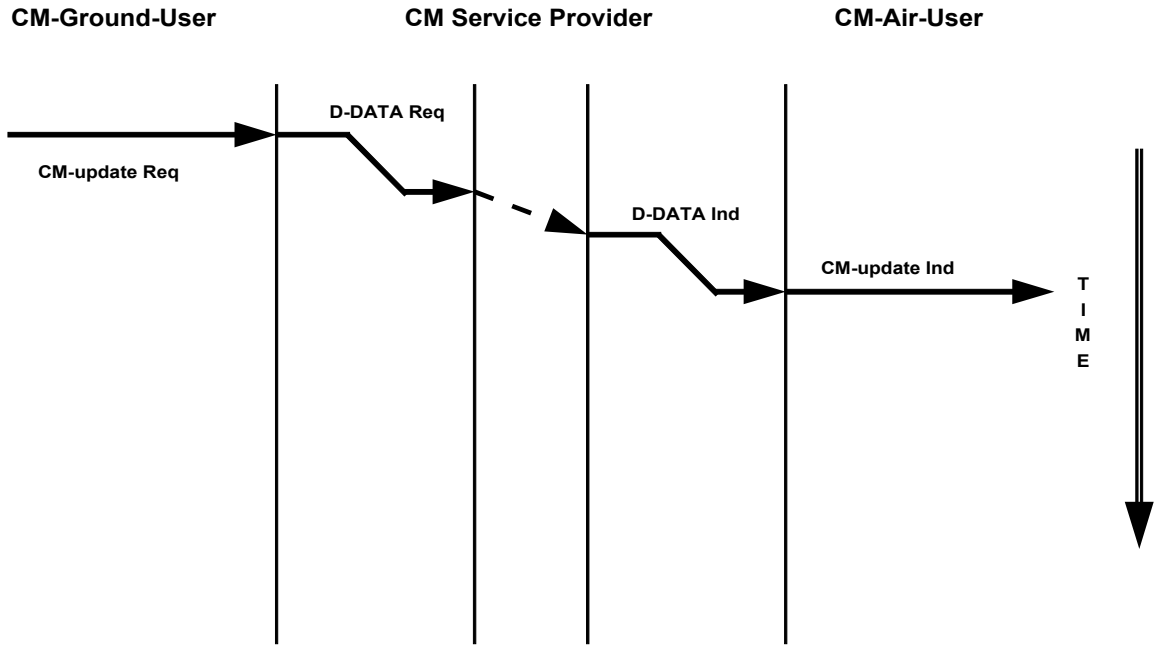


Figure 2.5-4a. Sequence Diagram for CM-update Service
Existing CM Dialogue

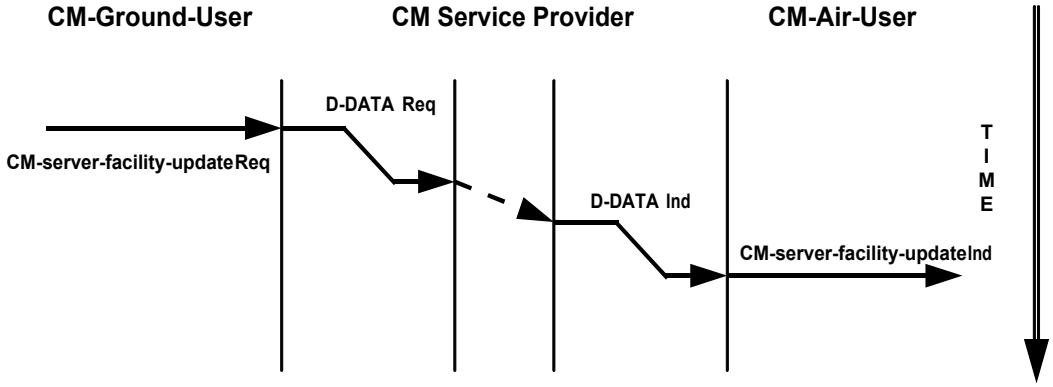


Figure 2.5-4b. Sequence Diagram for CM-server-facility-update Service Existing CM Dialogue (CM Version 2)

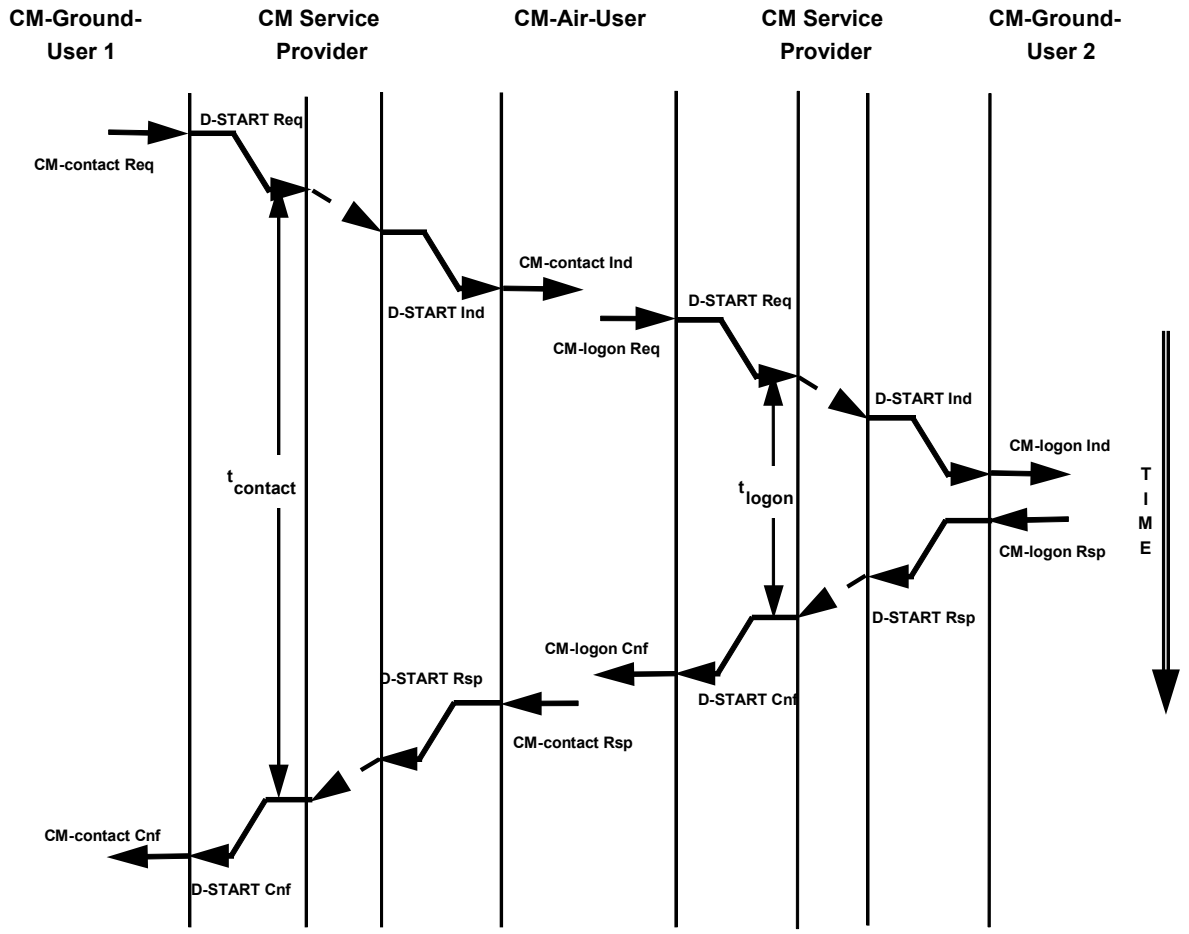


Figure 2.5-5. Sequence Diagram for CM-contact Service
No Existing CM Dialogue
With CM-logon Service as in Figure 2.5-1

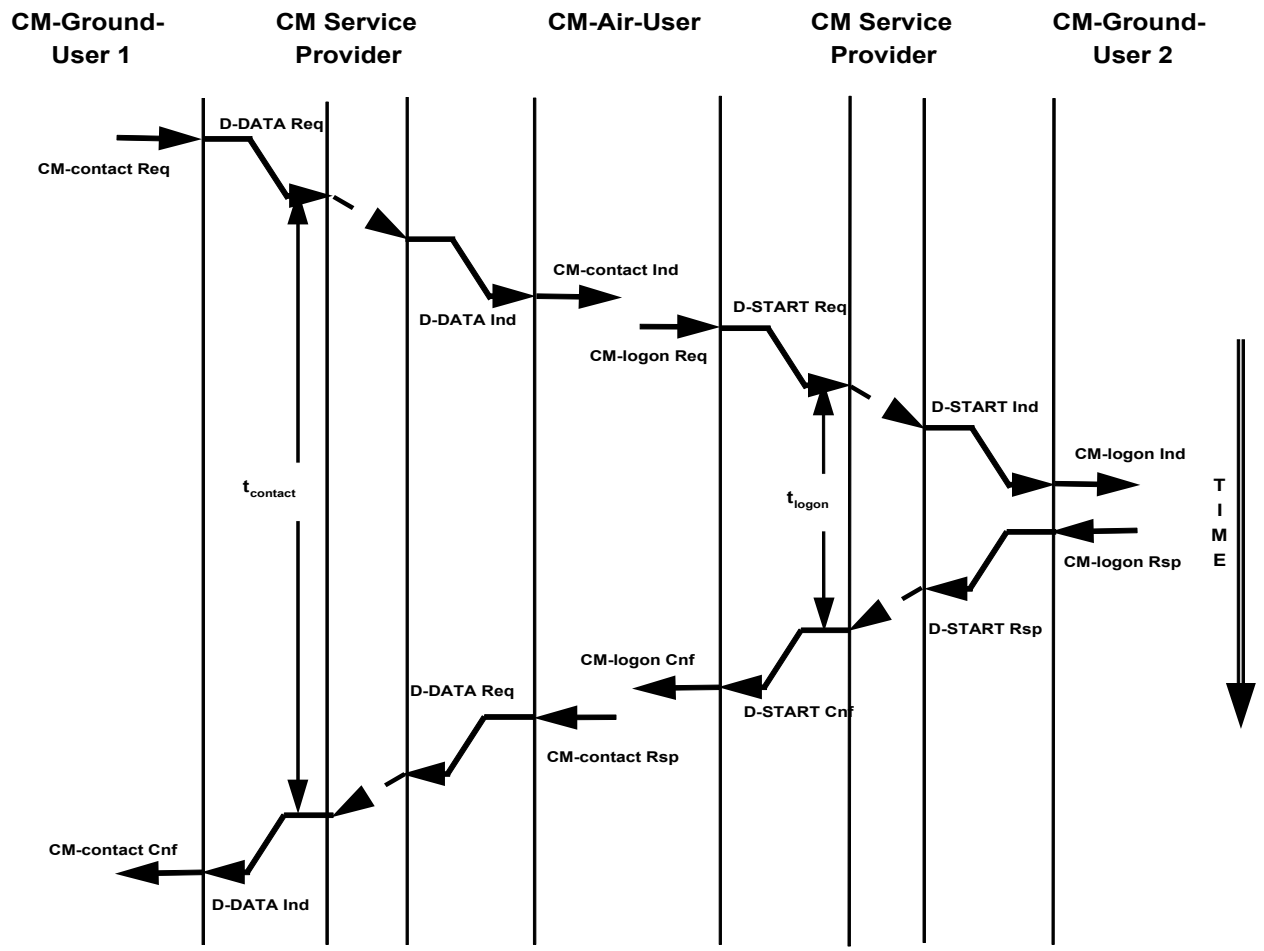


Figure 2.5-6. Sequence Diagram for CM-contact Service With Existing CM Dialogue With CM-logon Service as in Figure 2.5-1

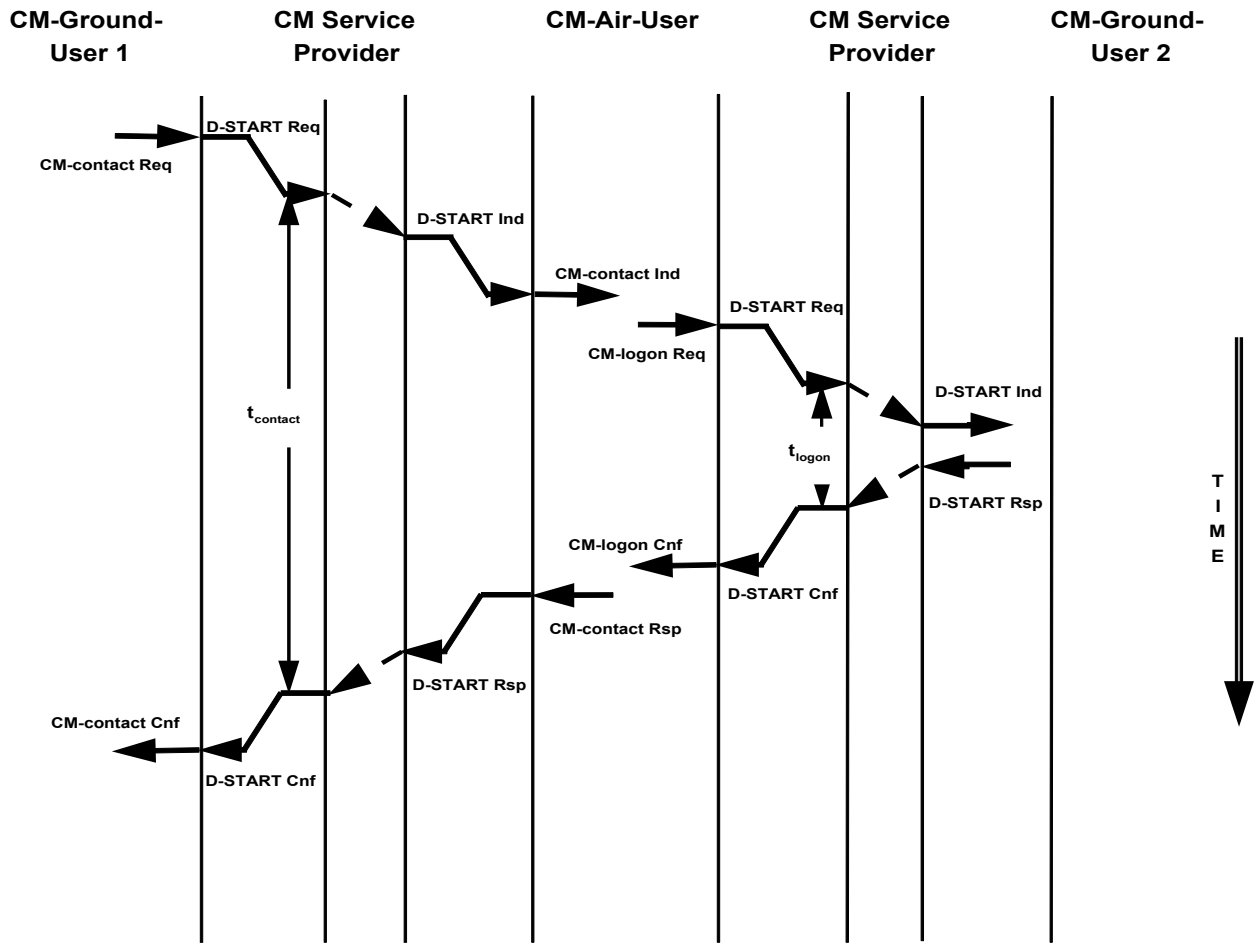


Figure 2.5-7. Sequence Diagram for CM-contact Service
No Existing CM Dialogue
With CM-logon Service as in Figure 2.5-2

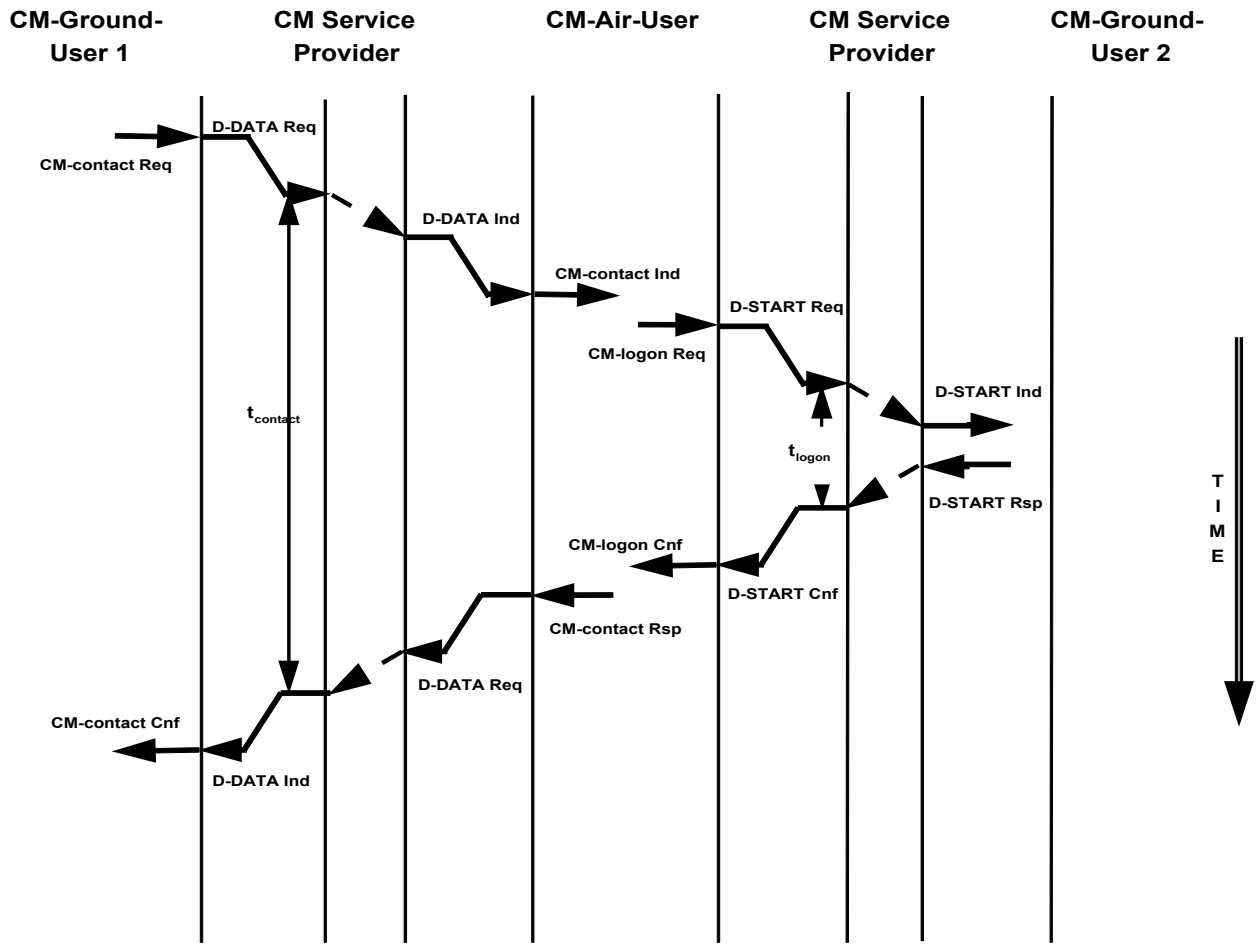


Figure 2.5-8. Sequence Diagram for CM-contact Service
Existing CM Dialogue
With CM-logon Service as in Figure 2.5-2

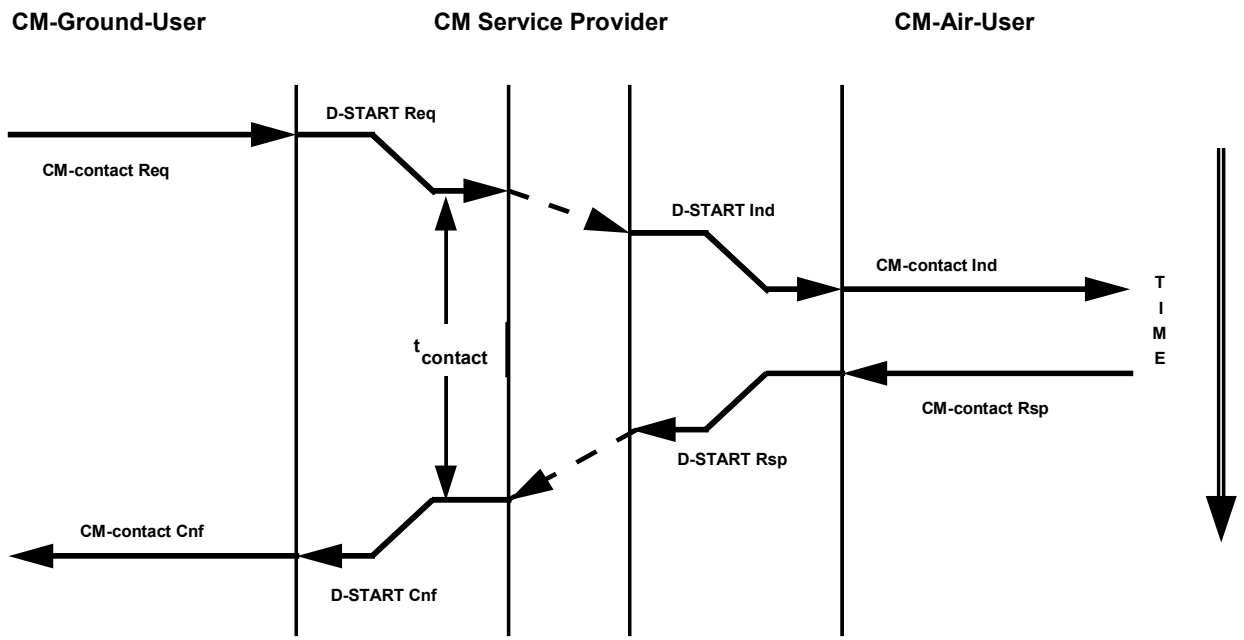


Figure 2.5-9. Sequence Diagram for CM-contact Service
No Existing CM Dialogue
Without CM-logon Service as Requested

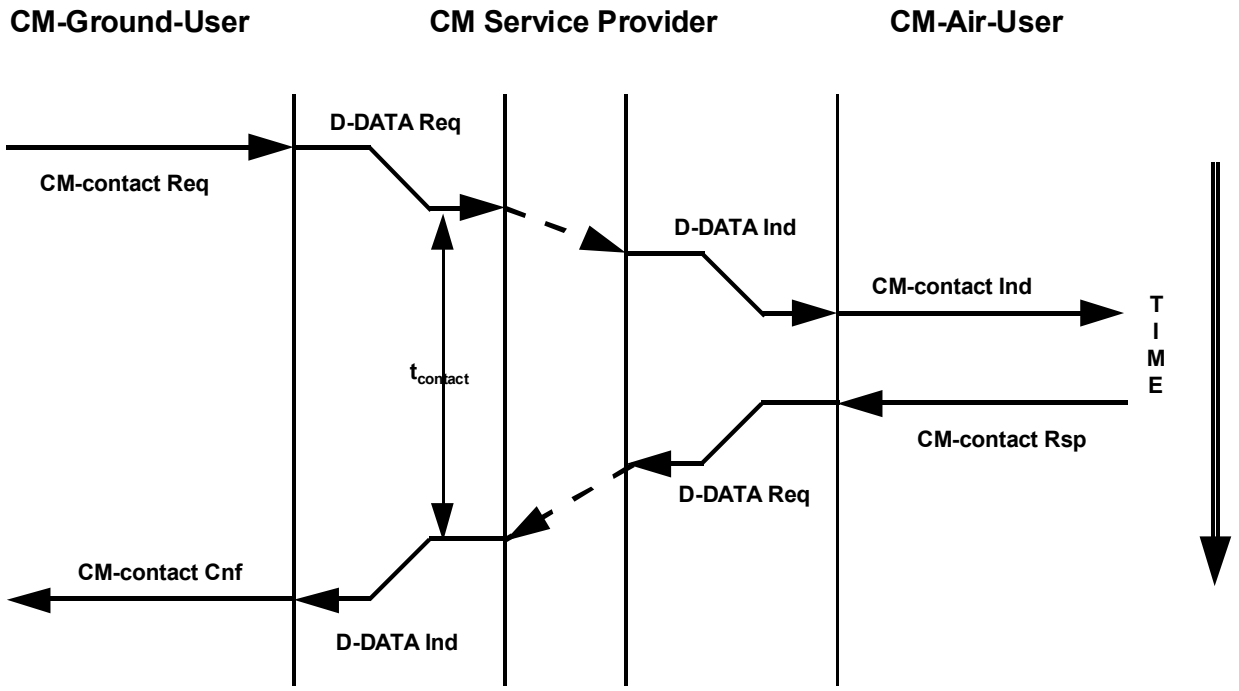


Figure 2.5-10. Sequence Diagram for CM-contact Service
With Existing CM Dialogue
Without CM-logon Service as Requested

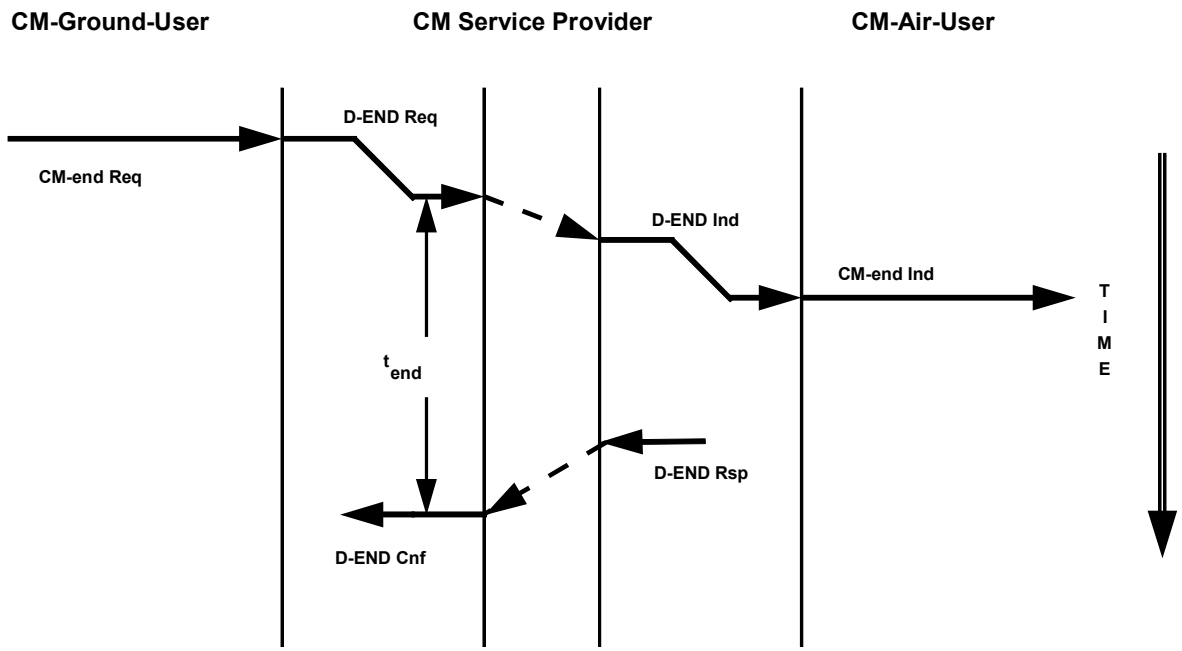


Figure 2.5-11. Sequence Diagram for CM-end Service

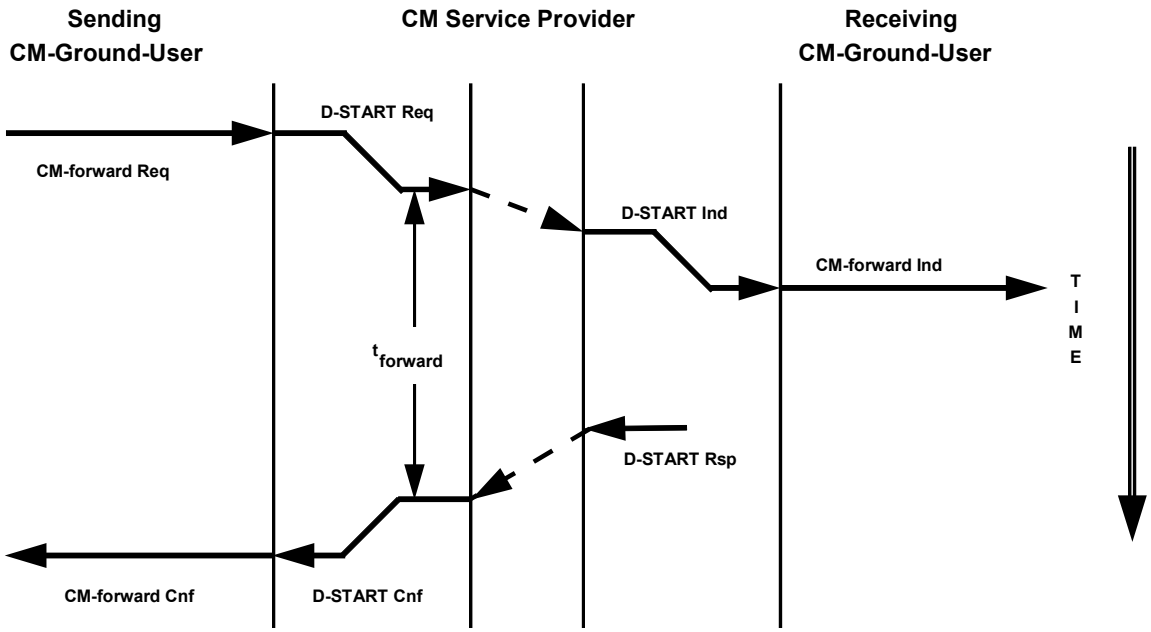


Figure 2.5-12. Sequence Diagram for CM-forward Service
Sending CM-Ground-ASE Version ≤ Receiving CM-Ground-ASE Version,
Ground-ground Forwarding Supported

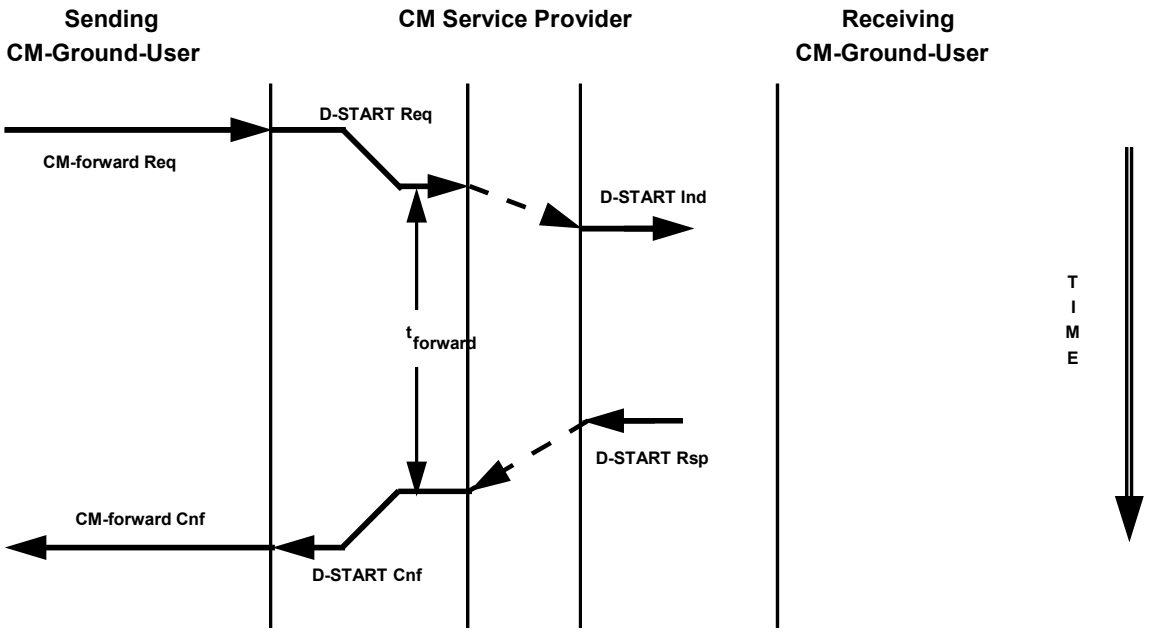


Figure 2.5-13. Sequence Diagram for CM-forward Service
Sending CM-Ground-ASE Version > Receiving CM-Ground-ASE Version
or Receiving CM-Ground-ASE does not Support Ground-Ground Forwarding

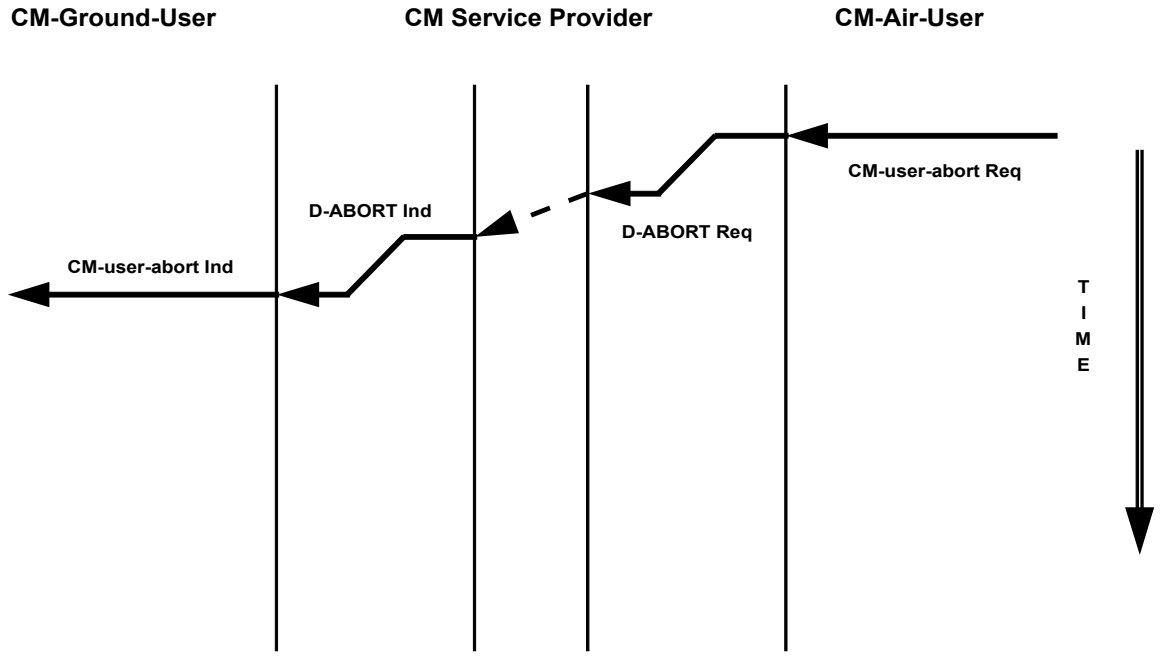


Figure 2.5-14. Sequence Diagram for CM-user-abort Service: CM-Air-User Initiated

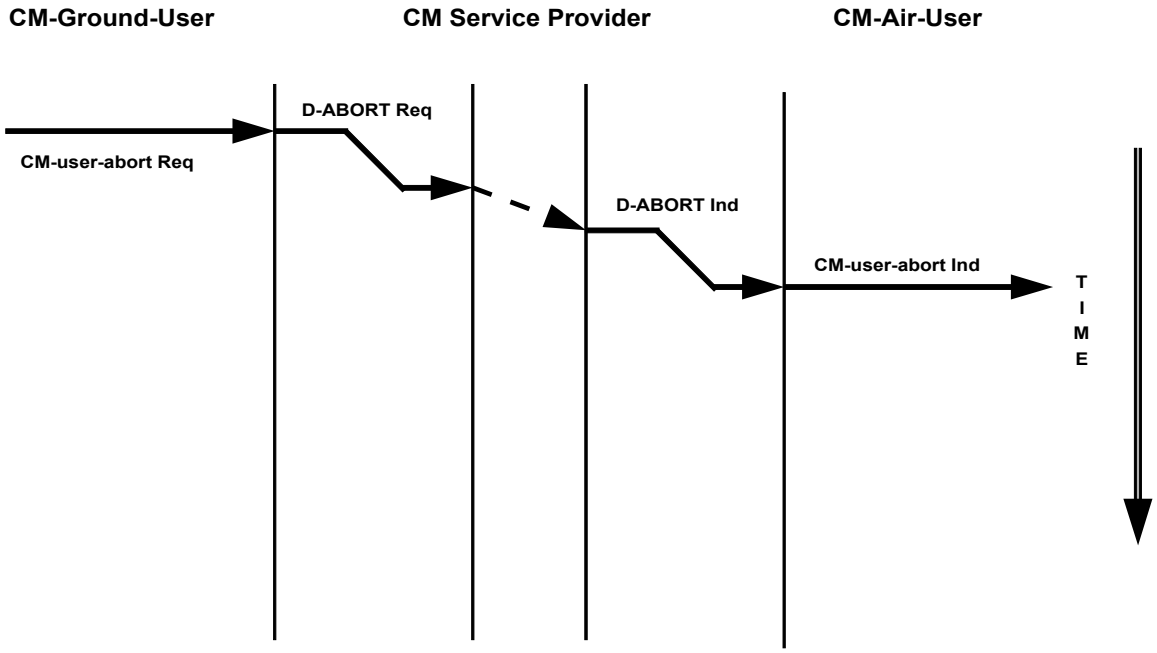


Figure 2.5-15. Sequence Diagram for CM-user-abort Service CM-Ground-User Initiated

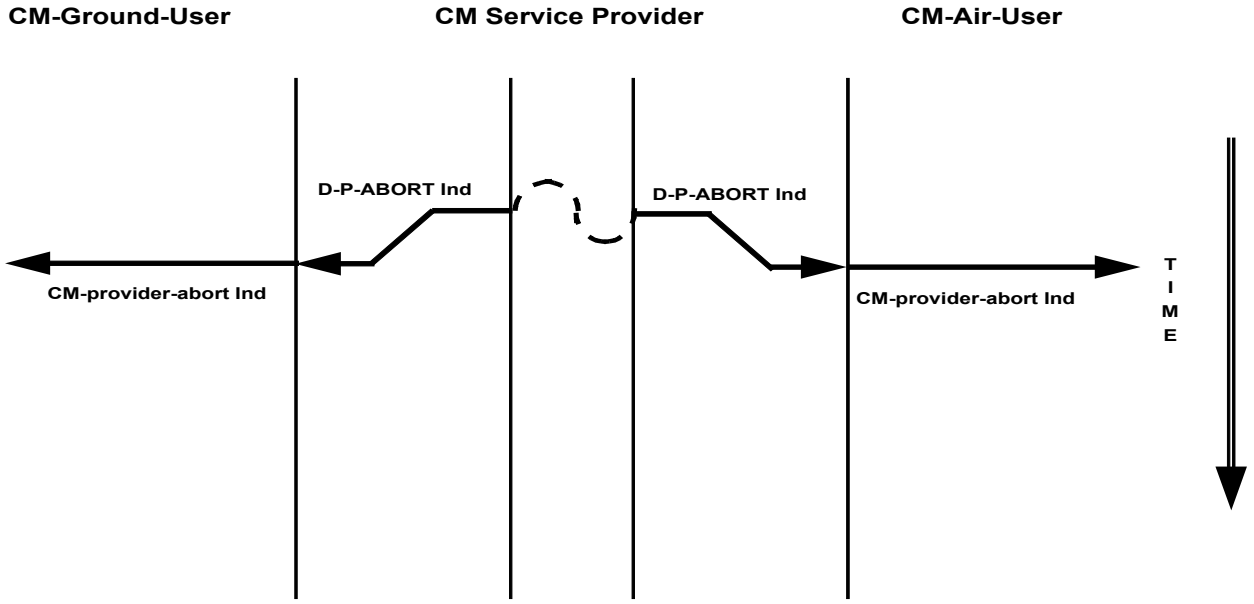


Figure 2.5-16. Sequence Diagram for CM-provider-abort Service: Dialogue Service Abort

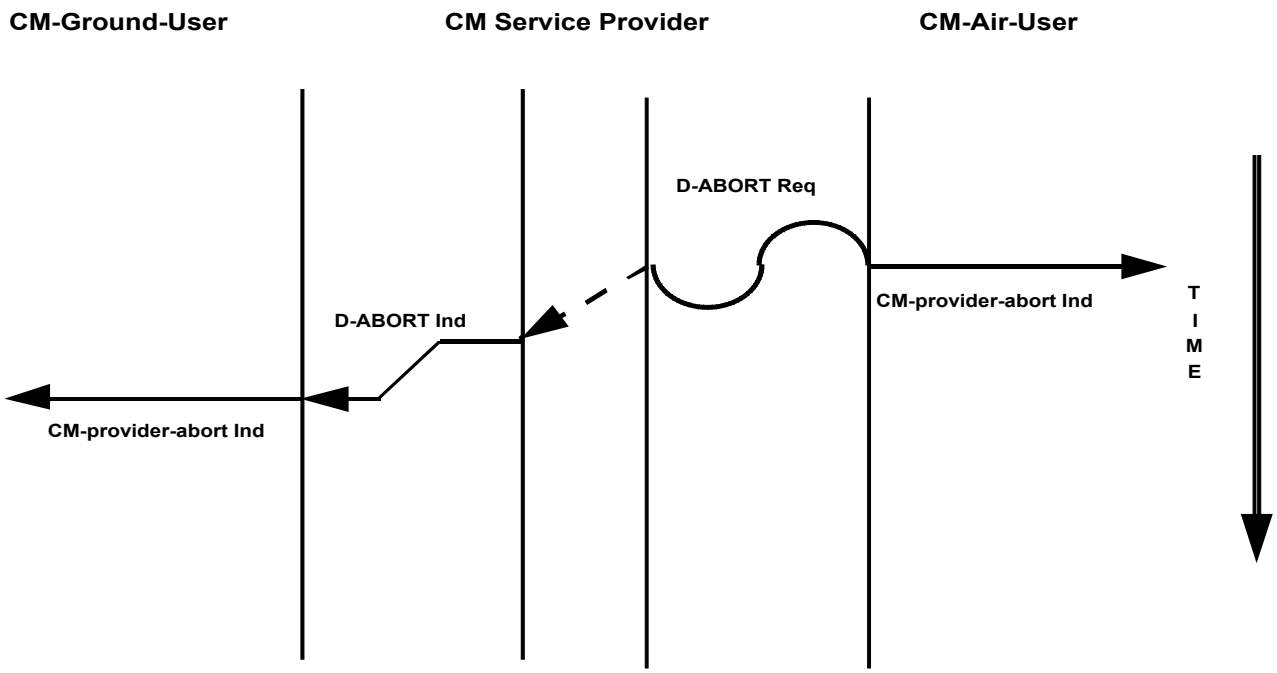


Figure 2.5-17. Sequence Diagram for CM-provider-abort Service: CM-Air-ASE Abort

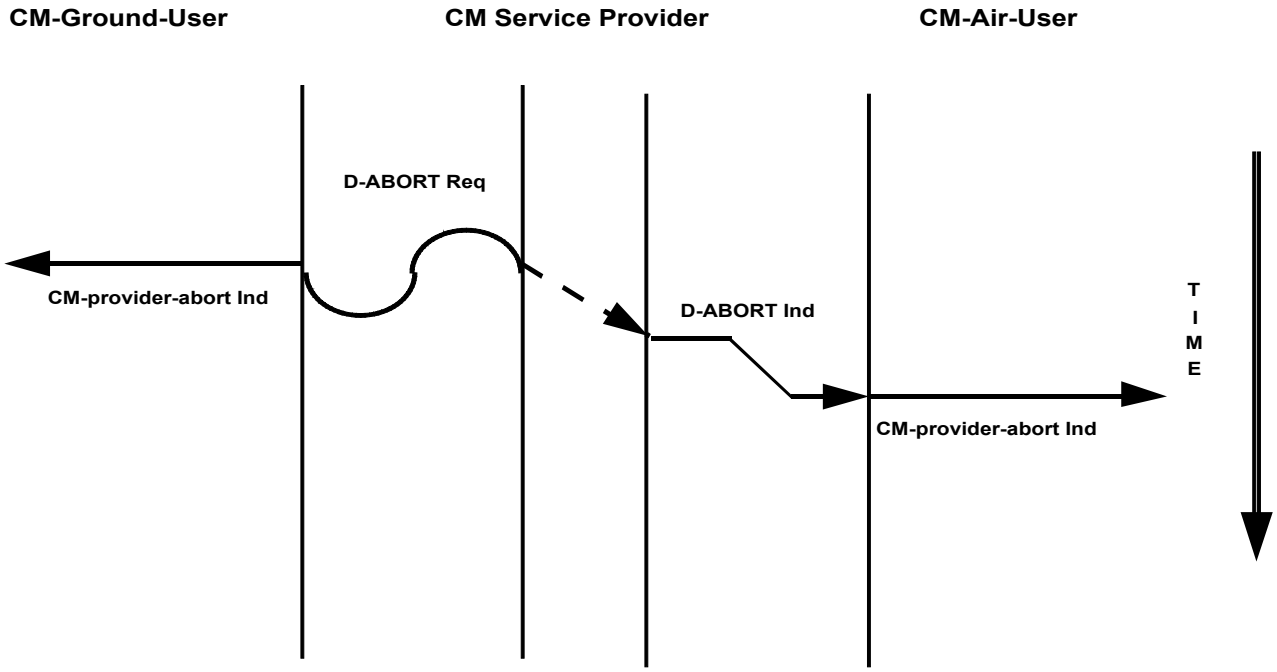


Figure 2.5-18. Sequence Diagram for CM-provider-abort Service: CM-Ground-ASE Abort

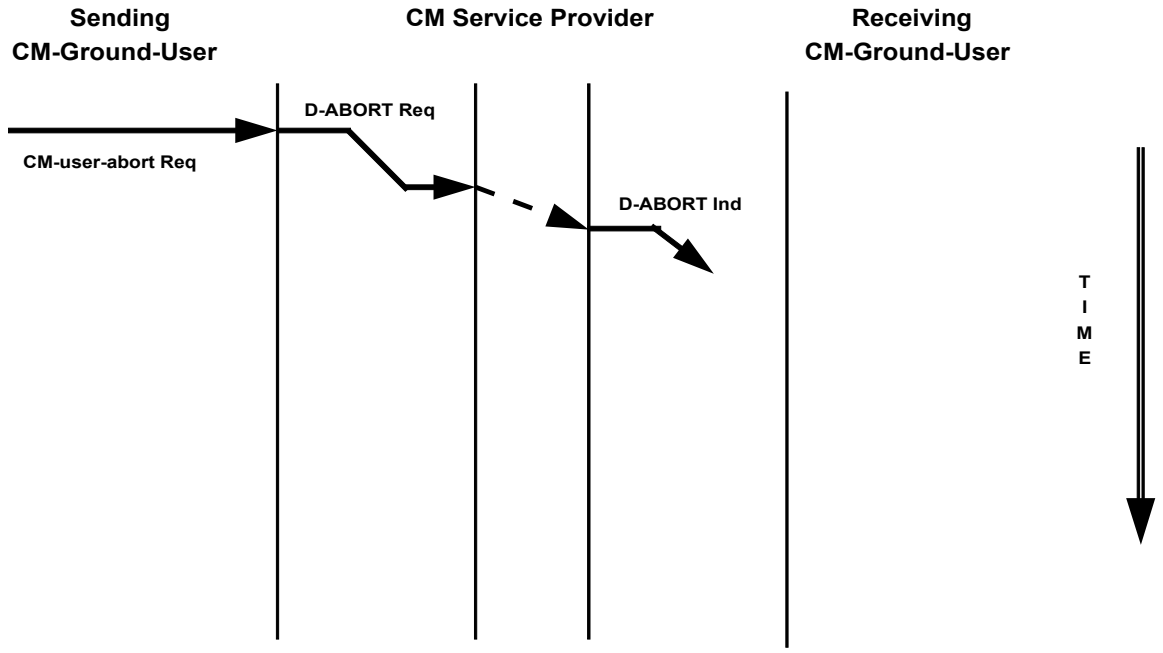


Figure 2.5-19. Sequence Diagram for CM-user-abort Service Sending CM-Ground-User Initiated

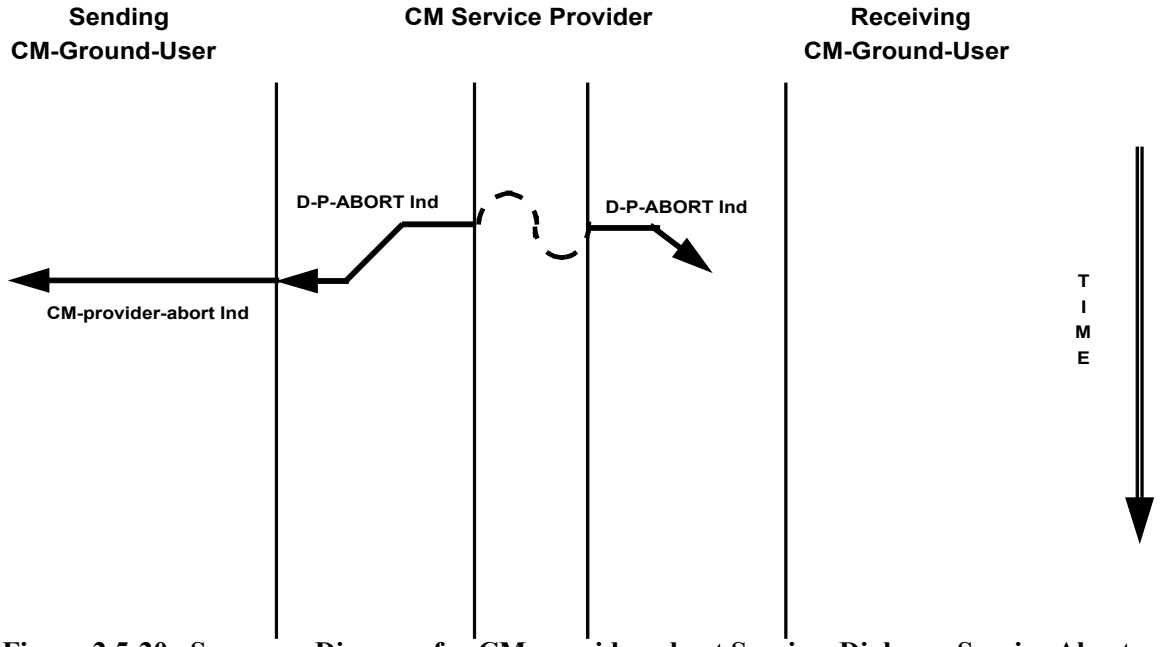


Figure 2.5-20. Sequence Diagram for CM-provider-abort Service: Dialogue Service Abort

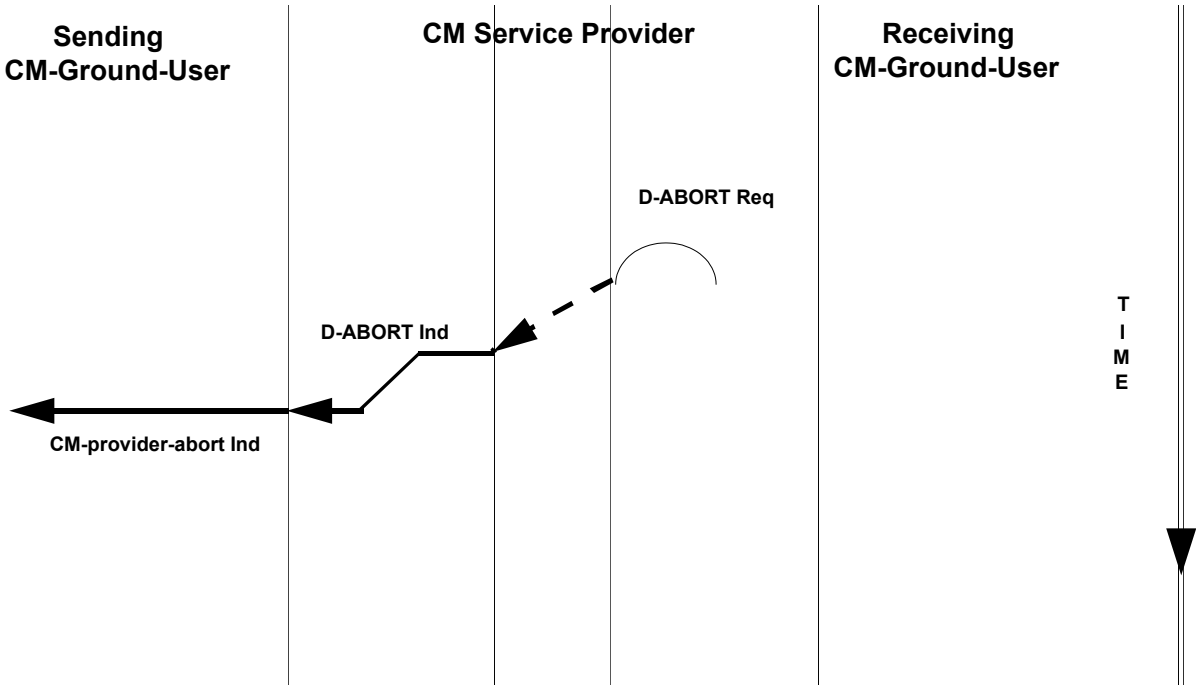
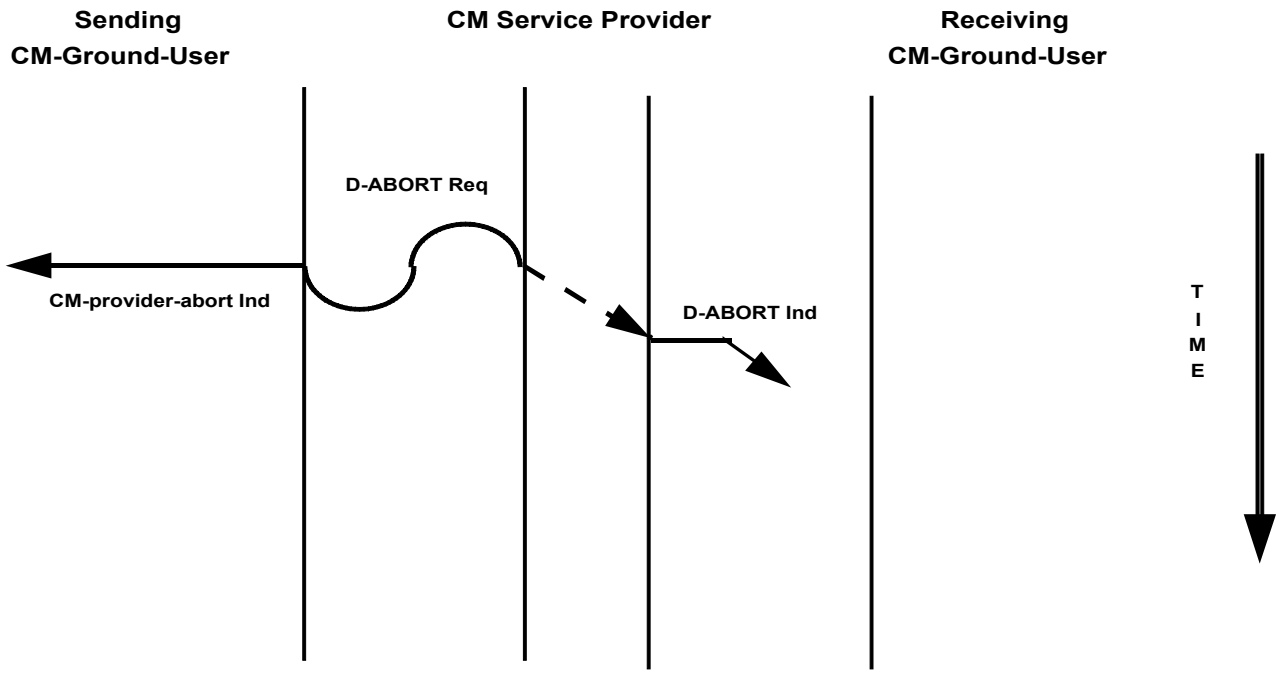


Figure 2.5-21: **Sequence Diagram for CM-provider-abort Service: Receiving CM-Ground-ASE Abort**

Figure 2.5-22. Sequence Diagram for CM-provider-abort Service: Sending CM-Ground-ASE Abort



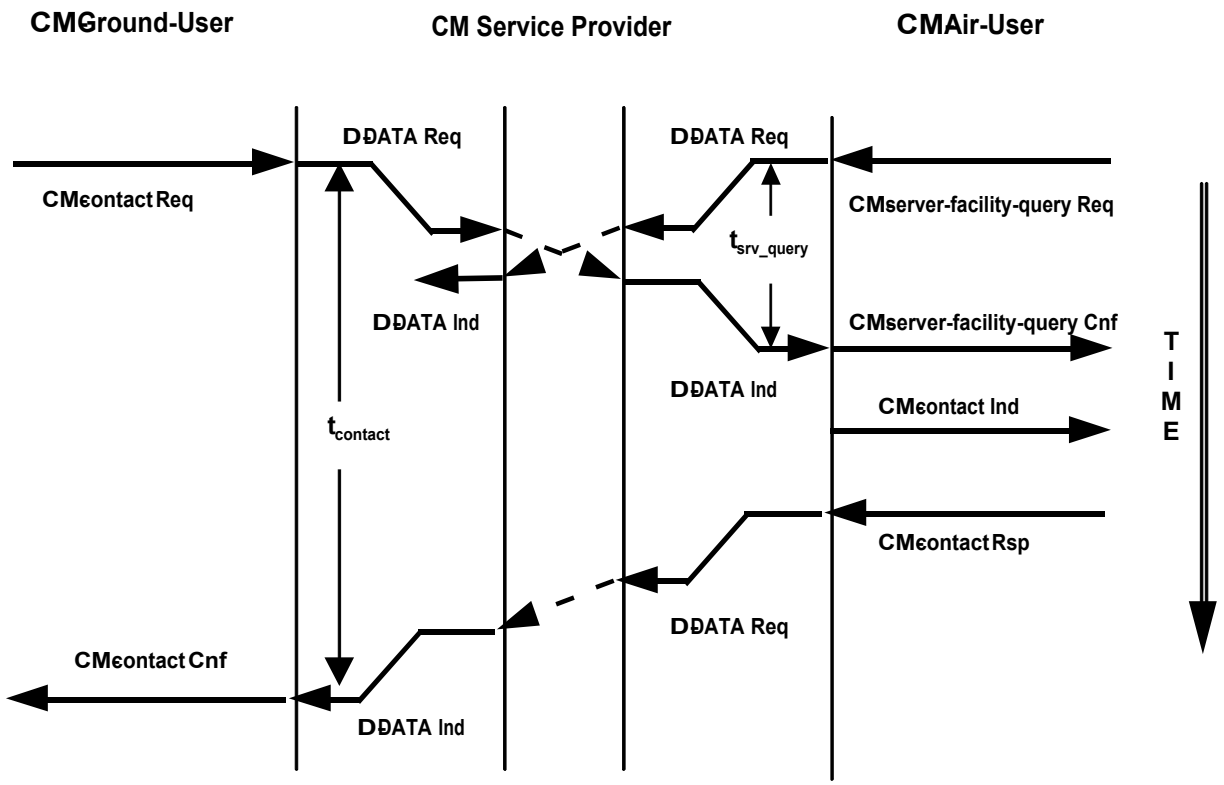


Figure 2.5-23. Collision of CM-contact and CM-server-facility-query

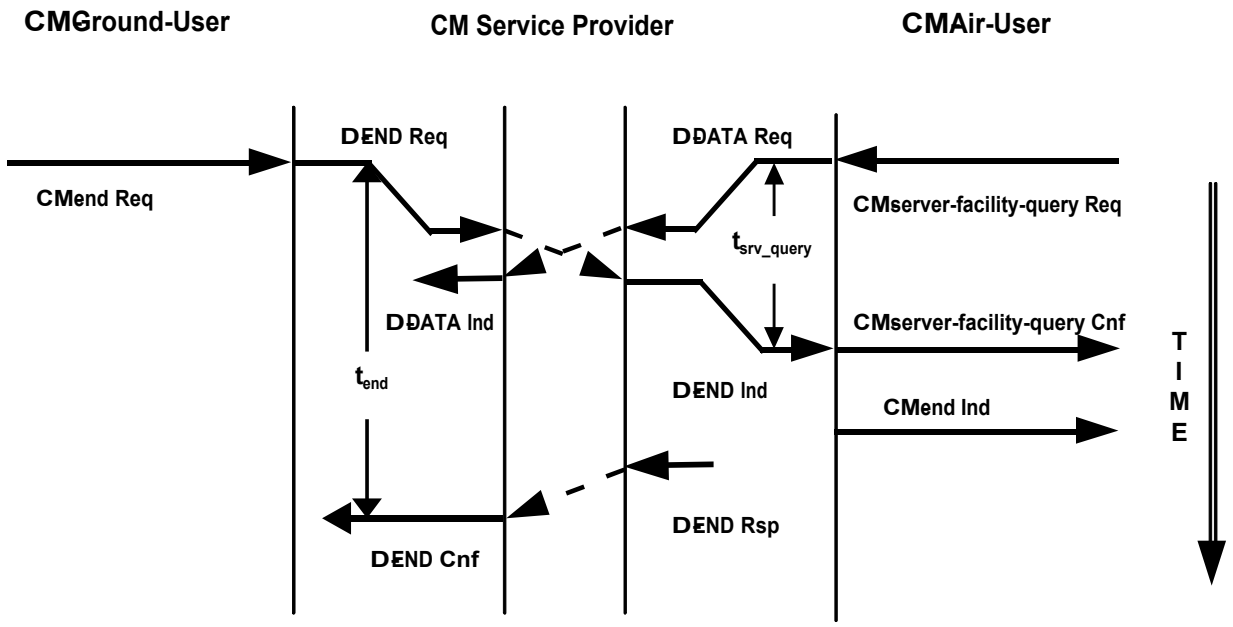


Figure 2.5-24. Collision of CM-end and CM-server-facility-query (CM Version 2)

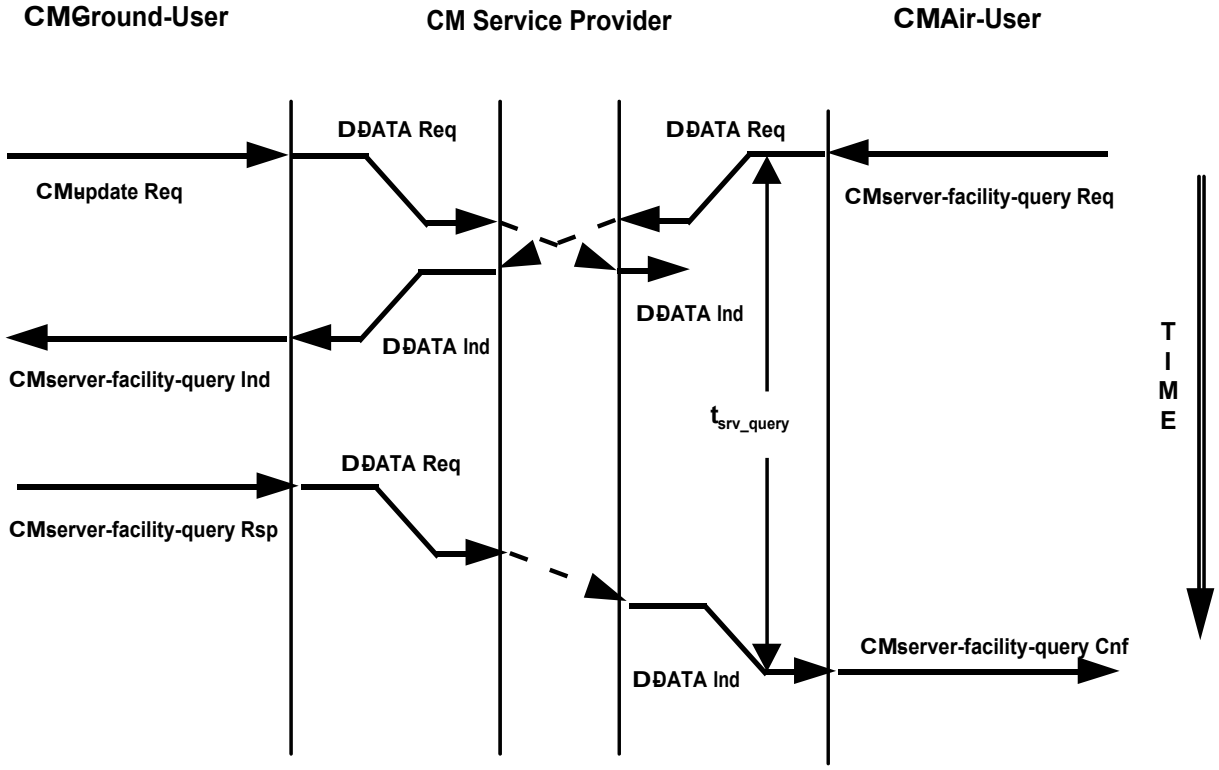


Figure 2.5-25. Collision of CM-update and CM-server-facility-query (CM Version 2)

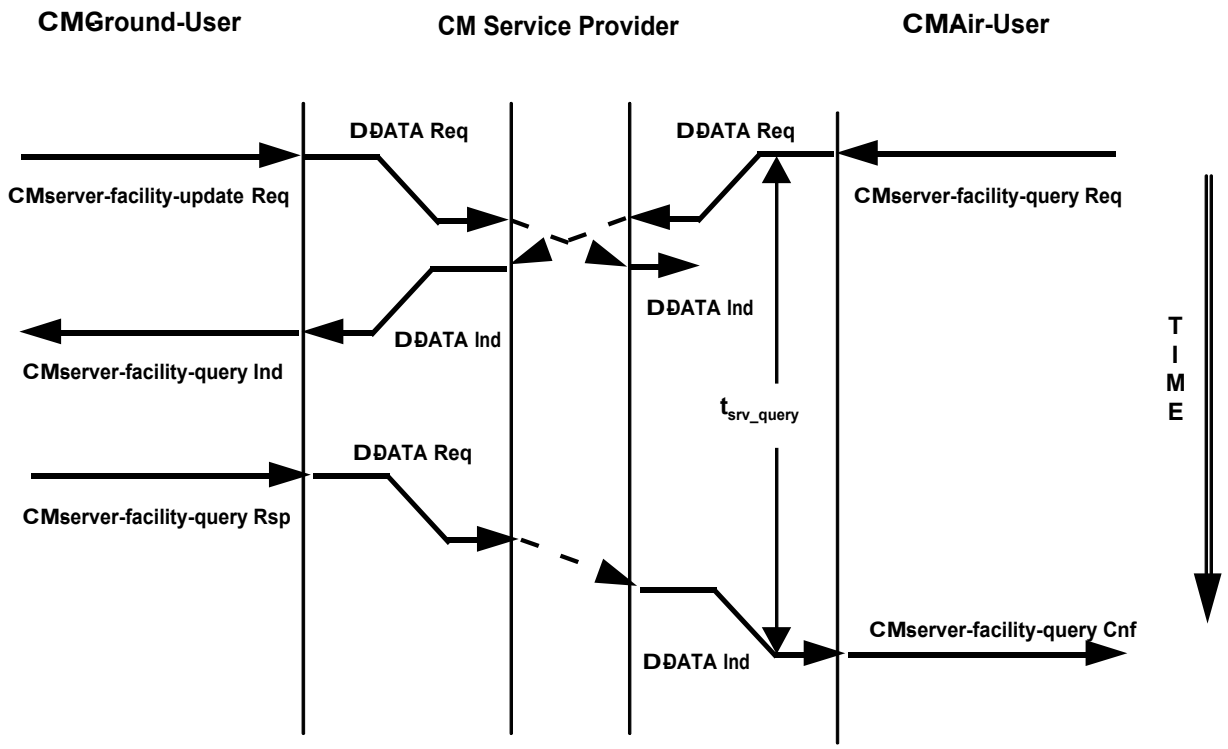


Figure 2.5-26. Collision of CM-server-facility-update and CM-server-facility-query (CM Version 2)

2.5.2 CM Service Provider Timers

2.5.2.1 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 version 1 and 2 applications. Table 2.5-1a lists the time constraints related only to the CM version 2 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 2.5.4.1.

2.5.2.2 **Recommendation**. —*The timer values should be as indicated in Table 2.5-1 for CM versions 1 and 2 and Table 2.5-1a for CM version 2.*

Table 2.5-1. CM Service Provider Timers for CM Versions 1 and 2

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

Table 2.5-1a. CM Service Provider Timers Unique to CM Version 2

CM Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
CM-server-facility-query	t_{srv_query}	6 min	D-START request, D-DATA request	D-START confirmation, D-DATA indication
CM-server-facility-update	t_{update}	4 min	D-START request	D-START confirmation

Note 2.— The receipt of a CM-user-abort request, D-ABORT indication, or D-P-ABORT indication are also timer stop events.

2.5.3 CM-ASE Protocol Description

2.5.3.1 Introduction

Note.— 2.5.3 presents requirements for CM-ASEs in specific states. 2.5.5 contains state tables for the CM-ASEs.

2.5.3.1.1 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.

2.5.3.1.2 Upon receipt of a PDU or dialogue service primitive, if no actions are described for their arrival when a CM-ASE is in a specific state, then they are considered not permitted and exception handling procedures as described in 2.5.4.4 shall apply.

2.5.3.1.3 If a PDU is received that cannot be decoded, then that PDU is considered an invalid PDU and exception handling procedures as described in 2.5.4.3 shall apply.

2.5.3.1.4 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 2.5.4.3 shall apply.

2.5.3.2 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) SERVER QUERY (CM version 2),

e) SERVER QUERY DIALOGUE (CM version 2),

f) DIALOGUE, and

g) 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 Ind, 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 with the ground system which invoked the CM-contact service,

c) in version 2, the CM-air-user invokes a CM-server-facility-query Req until it:

1) receives a CM-server-facility-query Cnf, if a dialogue is not maintained,

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

3) receives a CM-user abort,

4) receives a CM-provider abort, or

5) invokes a CM-user abort.

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

2.5.3.2.2 D-START Indication

2.5.3.2.2.1 Upon receipt of a D-START indication, if the CM-air-ASE is in the *IDLE* state and the D-START *Priority* QOS parameter has the abstract value "flight regularity communications", the D-START *RER* QOS parameter has the abstract value of "low", the D-START *Routing Class* QOS parameter identifies the traffic category "Air Traffic Service Communications (ATSC)", the *Calling Peer ID* parameter is a valid four to eight character facility designation and, if CM version 2, the D-START *Security Requirements* parameter is consistent with the local security policy, then:

2..5.3.2.2.1.1 If the APDU contained in the D-START User Data parameter is a CMGroundMesssage[CMUpdate] or, if CM version 2, a CMGroundMessage[CMSecureUpdate] 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, if CM version 2, the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication, and

c) enter the *IDLE* state.

2.5.3.2.2.1.2 If the APDU contained in the D-START *User Data* parameter is a CMGroundMessage[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.

2.5.3.2.2.1.3 For CM version 2, if the APDU contained in the D-START *User Data* is a CMGroundMessage[CMServerFacilityUpdate] APDU the CM-air-ASE shall:

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

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

2) the APDU contained in the D-START User Data parameter as the CM-server-facility-update Server Facility Update Information parameter value, and

b) invoke D-START response with the abstract value "rejected(permanent)" provided as D-START Result parameter and the D-START Security Requirements parameter value set to the same value as was received in the D-START indication, and

c) enter the IDLE state.

2.5.3.2.3 D-START Confirmation

2.5.3.2.3.1 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 either a CMGroundMessage-[CMLogonResponse] APDU or, if CM version 2, a CMGroundMessage[CMSecureLogonResponse] APDU and, if CM version 2, the D-START confirmation Security Requirements parameter value is the same as was set for the D-START request, 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 abstract value of “maintain” as the CM-Logon *Maintain Dialogue* parameter value, and

2) enter the *DIALOGUE* state.

Note.— For CM version 2, if a secure D-START request was issued (i.e. the Security Requirements *parameter was set to "Secured Dialogue supporting key management") then the D-START confirmation* Security Requirements *parameter must be equal to that value. If an unsecure D-START request was issued (i.e. the* Security Requirements *parameter was set to "No security") then the D-START confirmation* Security Requirements *parameter will be equal to that value. A CM version 1 peer will not set the* Security Requirements *parameter. However, the local upper layers will convert the absent parameter into a "No security" value. Therefore, the check to see that the received D-START confirmation* Security Requirements *parameter value is the same as the D-START request* Security Requirements *parameter value will work for CM version 2 communicating with both CM version 2 and CM version 1 peers.*

2.5.3.2.3.2 For CM version 2, upon receipt of a D-START confirmation, if the CM-air-ASE is in the *SERVER QUERY* state and if the APDU contained in the D-START *User Data* parameter is a CMGroundMessage[CMServerFacilityQueryResponse] APDU and the D-START confirmation *Security Requirements* parameter value is the same as was set for the D-START request, the CM-air-ASE shall:

a) stop timer t_{srv_query},

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-server-facility-query service confirmation with the APDU contained in the D-START *User Data* parameter as the CM-server-facility-query *Server Facility Query Response* parameter, and

2) enter the *IDLE* state.

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

1) invoke CM-server-facility-query service confirmation with the following:

i) the APDU contained in the D-START *User Data* parameter as the CM-server-facility-query Server Facility Query Response parameter,

ii) the abstract value of "maintain" as the CM-server-facility-query *Maintain Dialogue* parameter value, and

2) enter the *DIALOGUE* state.

2.5.3.2.4 D-DATA Indication

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

2.5.3.2.4.1.1 If the APDU contained in the D-DATA *User Data* parameter is a CMGroundMessage[CMUpdate] or, if CM version 2, a CMGroundMessage[CMSecureUpdate] 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.

2.5.3.2.4.1.2 If the APDU contained in the D-DATA *User Data* parameter is a CMGroundMessage[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.

2.5.3.2.4.1.3 For CM version 2, if the APDU contained in the D-DATA *User Data* parameter is a CMGroundMessage[CMServerFacilityUpdate] APDU the CM-air-ASE shall:

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

b) remain in the *DIALOGUE* state.

2.5.3.2.4.1.4 For CM version 2, upon receipt of a D-DATA indication, if the CM-air-ASE is in the *SERVER QUERY DIALOGUE* state, then:

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

a) stop timer t_{srv_query},

b) invoke CM-server-facility-query service confirmation with the APDU contained in the D-DATA *User Data* parameter as the CM-server-facility-query service *Server Facility Query Response* parameter value, and

c) enter the *DIALOGUE* state.

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

a) stop timer t_{srv_query},

b) invoke CM-server-facility-query service confirmation with the CMServerFacilityQueryResponse [InfoUnavailable:serviceInterrupted] APDU message element as the CM-server-facility-query service *Server Facility Query Response* parameter value,

c) 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

d) enter the *CONTACT DIALOGUE* state.

Note.— This is the collision case for a CM-server-facility-query service and a CM-contact service. The CM-air-user will abandon its CM-server-facility-query service and proceed with the CM-contact service.

2.5.3.2.4.2.2 If the APDU contained in the D-DATA *User Data* parameter is a CMGroundMessage[CMUpdate] APDU or a CMGroundMessage[CMServerFacilityUpdate] APDU, the CM-air-ASE shall:

a) discard the D-DATA indication, and

b) remain in the *SERVER QUERY DIALOGUE* state.

Note.— This is the collision case for a CM-server-facility-query service and a CM-update or CM-server-facility-update service. In either of these cases, if the aircraft gave precedence to the CM-update or CM-server-facility-update services (as it did for the CM-contact collision case) and abandons the CM-server-facility-query service, then the ground system will be left in a position where it may still respond to the CM-server-facility-query service after the CM-air-ASE has already completed it. This would result in a protocol error. Therefore, the CM-air-ASE ignores the CM-update or CM-server-facility-update services in this collision case.

2.5.3.2.5 D-END Indication

2.5.3.2.5.1 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.

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

a) stop timer t_{srv_query},

b) invoke CM-server-facility-query service confirmation with the CMServerFacilityQueryResponse [InfoUnavailable:serviceInterrupted] APDU message element as the CM-server-facility-query service *Server Facility Query Response* parameter value,

c) invoke CM-end service indication,

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

e) enter the *IDLE* state.

Note.— This is the collision case between the CM-server-facility-query service and the CM-end service. The CM-air-ASE will abandon its CM-server-facility-query service and proceed with the CM-end service.

2.5.3.2.6 CM-logon and CM-server-facility-query Service Request

2.5.3.2.6.1 Upon receipt of a CM-logon service request:

2.5.3.2.6.1.1 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) for CM version 2, if the *Emulated Version* parameter is not provided, the CM-air-ASE version number as the D-START *DS User Version Number* parameter value, else the *Emulated Version* parameter value 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) for CM version 2, the CM-logon *Security Required* parameter value as the D-START *Security Requirements* parameter value,

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

c) start timer t_{logon}, and

d) enter the *LOGON* state.

2.5.3.2.6.2 For CM version 2, upon receipt of a CM-server-facility-query request:

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

a) create a CMAircraftMessage APDU with a cmServerFacilityQueryRequest APDU-element based on the *Server Facility Query Request* parameter value;

b) invoke D-START request with the following:

1) the CM-server-facility-query *Facility Designation* parameter value as the D-START *Called Peer ID* parameter value;

2) the CM-server-facility-query *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 CM-server-facility-query *Security Required* parameter value as the D-START *Security Requirements* parameter value;

5) 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

6) the CMAircraftMessage APDU as the D-START *User Data* parameter value;

c) start timer t_{srv_query};and

d) enter the *SERVER QUERY* state.

2.5.3.2.6.2.2 If the CM-air-ASE is in the *DIALOGUE* state the CM-air-ASE shall:

a) create a CMAircraftMessage APDU with a cmServerFacilityQueryRequest APDU-element based on the *Server Facility Query Request* parameter value;

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

c) enter the *SERVER QUERY DIALOGUE* state.

2.5.3.2.7 CM-contact Service Response

2.5.3.2.7.1 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;

2) if CM version 2, the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication, and

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

c) enter the *IDLE* state.

2.5.3.2.7.2 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.

2.5.3.2.8 CM-user-abort Service Request

2.5.3.2.8.1 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} or, if CM version 2, t_{srv_query}, 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.

2.5.3.2.9 D-ABORT Indication

2.5.3.2.9.1 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} or, if CM version 2, t_{srv_query}, 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.

2.5.3.2.10 D-P-ABORT Indication

2.5.3.2.10.1 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} or, if CM version 2, t_{srv_query}, 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.

2.5.3.3 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) SERVER QUERY (CM version 2),

h) SERVER QUERY DIALOGUE (CM version 2),

i) END, and

j) 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,

d) for version 2, the CM-ground-user receives a CM-server-facility-query Ind until it:

1) invokes a CM-server-facility-query 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.

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

2.5.3.3.2 D-START Indication

2.5.3.3.2.1 Upon receipt of a D-START indication, if the CM-ground-ASE is in the *IDLE* state and the abstract value of the D-START *Calling Peer ID* parameter is the ICAO 24 bit aircraft address and the D-START *Priority* QOS parameter has the abstract value "flight regularity communications", the D-START *RER* QOS parameter has the abstract value of "low" and the D-START *Routing Class* QOS parameter identifies the traffic category "Air Traffic Service Communications (ATSC)" and, if CM version 2, the D-START *Security Requirements* parameter is consistent with the local security policy, then:

2.5.3.3.2.1.1 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.

Note.— For CM version 2, the D-START Security Requirements *parameter is not explicitly set by the CM-ground-ASE in the case that the aircraft's CM version number is greater than the ground's CM version number.*

2.5.3.3.2.1.2 If the D-START *DS User Version Number* parameter value is less than the CM-ground-ASE version number and the APDU contained in the D-START *User Data* parameter is a CMAircraftMessage[CMLogonRequest] APDU, 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) if CM version 2, the D-START *Security Requirements* parameter value as the CM-logon service *Security Required* parameter value,

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

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

b) enter the *LOGON* state.

2.5.3.3.2.1.3 If the D-START *DS User Version Number* parameter value is equal to CM-ground-ASE version number and the APDU contained in the D-START *User Data* parameter is a CMAircraftMessage[CMLogonRequest] APDU, 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,

2) if CM version 2, the D-START *Security Requirements* parameter value as the CM-logon service *Security Required* 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

2.5.3.3.2.2 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 CMGroundMessage-[CMForwardRequest] or, if CM version 2, a CMGroundMessage[CMEnhancedForwardRequest] 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 *Priority* QOS parameter has the abstract value "flight regularity communications" and the D-START *RER* QOS parameter has the abstract value of "low"and the D-START *Routing Class* QOS parameter identifies the traffic category "Air Traffic Service Communications (ATSC)", then:

2.5.3.3.2.2.1 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.

Note.— For CM version 2, the D-START Security Requirements *parameter is not explicitly set by the CM-ground-ASE if the CM-forward service is not supported.*

2.5.3.3.2.2.2 If the D-START *DS User Version Number* parameter value is greater than the CM-ground-ASE version number and, if CM version 2, the D-START *Security Requirements* parameter is consistent with the local security policy 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,

4) if CM version 2, the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication, and

c) remain in the *IDLE* state.

2.5.3.3.2.2.3 If the D-START *DS User Version Number* parameter value is less than the CM-ground-ASE version number and, if CM version 2, the D-START *Security Requirements* parameter is consistent with the local security policy 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,

2) if CM version 2, the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication, and

3) 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.

2.5.3.3.2.2.4 If the D-START *DS User Version Number* parameter value is equal to the CM-ground-ASE version number and, if CM version 2, the D-START *Security Requirements* parameter is consistent with the local security policy 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,

2) if CM version 2, the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication, and

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

d) remain in the *IDLE* state.

2.5.3.3.2.3 For CM version 2, 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 CMAircraftMessage[CMServerFacilityQueryRequest] APDU and the abstract value of the D-START *Calling Peer ID* parameter is the ICAO 24 bit aircraft address and the D-START *Priority* Quality-of-Service parameter has the abstract value "flight regularity communications" and the *RER* Quality-of-Service parameter has the abstract value "low" and the D-START *Security Requirements* parameter is consistent with the local security policy, then the CM-ground-ASE shall:

a) invoke CM-server-facility-query indication with:

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

2) the D-START *Security Requirements* parameter value as the CM-server-facility-query service *Security Required* parameter value, and

3) the APDU in the D-START *User Data* parameter as the CM-server-facility-query service *Server Facility Query Request* parameter value, and

b) enter the *SERVER QUERY* state.

2.5.3.3.3 D-START Confirmation

2.5.3.3.3.1 Upon receipt of a D-START confirmation:

2.5.3.3.3.1.1 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" and, if CM version 2, the abstract value of the D-START *Security Requirements* parameter is the same as was set in the D-START request, enter the *IDLE* state.

Note.— For CM version 2, if a secure D-START request was issued (i.e. the Security Requirements *parameter was set to "Secured Dialogue supporting key management" or "Secured Dialogue") then the D-START confirmation* Security Requirements *parameter must be equal to that value. If an unsecure D-START request was issued (i.e. the* Security Requirements *parameter was set to "No security") then the D-START confirmation* Security Requirements *parameter will be equal to that value. A CM version 1 peer will not set the* Security Requirements *parameter. However, the local upper layers will convert the absent parameter into a "No security" value. Therefore, the check to see that the received D-START confirmation* Security Requirements *parameter value is the same as the D-START request* Security Requirements *parameter value will work for CM version 2 communicating with both CM version 2 and CM version 1 peers.*

2.5.3.3.3.1.2 If the CM-ground-ASE is in the *CONTACT* state and the APDU contained in the D-START *User Data* parameter is a CMAircraftMessage[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" and, if CM version 2, the abstract value of the D-START *Security Requirements* parameter is the same as was set in the D-START request 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.

Note.— For CM version 2, if a secure D-START request was issued (i.e. the Security Requirements *parameter was set to "Secured Dialogue supporting key management" or "Secured Dialogue") then the D-START confirmation* Security Requirements *parameter must be equal to that value. If an unsecure D-START request was issued (i.e. the* Security Requirements *parameter was set to "No security") then the D-START confirmation* Security Requirements *parameter will be equal to that value. A CM version 1 peer will not set the* Security Requirements *parameter. However, the local upper layers will convert the absent parameter into a "No security" value. Therefore, the check to see that the received D-START confirmation* Security Requirements *parameter value is the same as the D-START request* Security Requirements *parameter value will work for CM version 2 communicating with both CM version 2 and CM version 1 peers.*

2.5.3.3.3.1.3 If the CM-ground-ASE is in the *FORWARD* state and if the D-START *User Data* parameter is a CMGroundMessage[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" and, if CM version 2, the abstract value of the D-START confirmation *Security Requirements* parameter is the same as was set in the D-START request, 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" and, if CM version 2, the abstract value of the D-START confirmation *Security Requirements* parameter is the same as was set in the D-START request, 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" and, if CM version 2, the abstract value of the D-START confirmation *Security Requirements* parameter is the same as was set in the D-START request, invoke a CM-forward

service confirmation with the CM-forward *Result* parameter set to the abstract value "success", and

e) enter the *IDLE* state.

Note.— For CM version 2, if a secure D-START request was issued (i.e. the Security Requirements *parameter was set to "Secured Dialogue supporting key management") then the D-START confirmation* Security Requirements *parameter must be equal to that value. If an unsecure D-START request was issued (i.e the* Security Requirements *parameter was set to "No security") then the D-START confirmation* Security Requirements *parameter will be equal to that value. A CM version 1 peer will not set the* Security Requirements *parameter. However, the local upper layers will convert the absent parameter into a "No security" value. Therefore, the check to see that the received D-START confirmation* Security Requirements *parameter value is the same as the D-START request* Security Requirements *parameter value will work for CM version 2 communicating with both CM version 2 and CM version 1 peers.*

2.5.3.3.4 D-DATA Indication

2.5.3.3.4.1 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 CMAircraftMessage[CMContactResponse] APDU, the CM-ground-ASE shall:

a) stop timer $t_{contact}$,

b) 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

c) enter the *DIALOGUE* state.

2.5.3.3.4.2 For CM version 2, upon receipt of a D-DATA indication, if the CM-ground-ASE is in the *DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a CMAircraftMessage[CMServerFacilityQueryRequest] APDU, the CM-ground-ASE shall:

a) invoke CM-server-facility-query service indication with the APDU contained in the D-DATA *User Data* parameter as the CM-server-facility-query *Server Facility Query Request* parameter value, and

b) enter the *SERVER QUERY DIALOGUE* state.

2.5.3.3.4.3 For CM version 2, 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 CMAircraftMessage[CMServerFacilityQueryRequest] APDU, the CM-ground-ASE shall:

a) discard the D-DATA indication, and

b) remain in the *CONTACT DIALOGUE* state.

Note.— This is the collision case for the CM-contact and CM-server-facility-query services. In this case, the CM-ground-ASE ignores the CM-server-facility-query indication, and continues with the CM-contact service. The peer aircraft with abandon the CM-server-facility-query service and proceed with the CM-contact service.

2.5.3.3.4.4 For CM version 2, upon receipt of a D-DATA indication, if the CM-ground-ASE is in the *END* state and the APDU contained in the D-DATA *User Data* parameter is a CMAircraftMessage[CMServerFacilityQueryRequest] APDU, the CM-ground-ASE shall:

a) discard the D-DATA indication, and

b) remain in the *END* state.

Note.— This is the collision case for the CM-end and CM-server-facility-query services. In this case, the CM-ground-ASE ignores the CM-server-facility-query indication, and continues with the CM-end service. The peer aircraft with abandon the CM-server-facility-query service and proceed with the CM-end service.

2.5.3.3.5 D-END Confirmation

2.5.3.3.5.1 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.

2.5.3.3.6 CM-logon and CM-server-facility-query Service Response

2.5.3.3.6.1 Upon receipt of a CM-logon service response, if the CM-ground-ASE is in the *LOGON* state and, if version 1 CM or if version 2 CM emulating a version 1 CM, 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.

2.5.3.3.6.2 For CM version 2 not communicating with a version 1 CM, 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 cmSecureLogonResponse 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) the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication,

3) 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.

4) 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.

Note.— The CM version number is not included in the D-START response primitive if the CM-ground-ASE version number is less than or equal to the CM-air-ASE version number.

2.5.3.3.6.3 For CM version 2, upon receipt of a CM-server-facility-query response:

2.5.3.3.6.3.1 If the CM-ground-ASE is in the *SERVER QUERY* state, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmServerFacilityQueryResponse APDU-element based on the CM-server-facility-query *Server Facility Query 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) the D-START *Security Requirements* parameter value set to the same value as was received in the D-START indication,

3) if the CM-server-facility-query *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.

4) if the CM-server-facility-query *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.

2.5.3.3.6.3.2 If the CM-ground-ASE is in the *SERVER QUERY DIALOGUE* state, the CM-ground-ASE shall:

a) create a CMGroundMessage APDU with a cmServerFacilityQueryResponse APDU-element based on the CM-server-facility-query *Server Facility Query Response* parameter value,

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

c) enter the *DIALOGUE* state.

2.5.3.3.7 CM-update and CM-server-facility-update Service Request

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

a) if version 1 CM, or if version 2 CM and the *Security Required* parameter is set to the abstract value "No Security", create a CMGroundMessage APDU with a cmUpdate APDU-element based on the CM-update *Update Information* parameter value,

b) if version 2 CM and the *Security Required* parameter is not set to the abstract value "No Security", create a CMGroundMessage APDU with a cmSecureUpdate APDU-element based on the CM-update *Update Information* parameter value,

c) 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,

5) if CM version 2, the CM-update *Security Required* parameter value as the D-START *Security Requirements* parameter value, and

d) start timer t_{update}, and

e) enter the *UPDATE* state.

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

a) if version 1 CM, or if version 2 CM emulating a version 1 CM, create a CMGroundMessage APDU with a cmUpdate APDU-element based on the CM-update *Update Information* parameter value,

b) if CM version 2 not emulating a version 1 CM, create a CMGroundMessage APDU with a cmSecureUpdate APDU-element based on the CM-update *Update Information* parameter value,

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

d) remain in the *DIALOGUE* state.

2.5.3.3.7.3 For CM version 2, upon receipt of a CM-server-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 cmServerFacilityUpdate APDU-element based on the CM-server-facility-update *Server Facility Update* parameter value,

b) invoke D-START request with the following:

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

2) the CM-server-facility-update *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-server-facility-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,

5) the CM-server-facility-update *Security Required* parameter value as the D-START *Security Requirements* parameter value, and

c) start timer t_{update}, and

d) enter the *UPDATE* state.

2.5.3.3.7.4 For CM version 2, upon receipt of a CM-server-facility-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 cmServerFacilityUpdate APDU-element based on the CM-server-facility-update *Server Facility Update* 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.

2.5.3.3.8 CM-contact Service Request

2.5.3.3.8.1 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) if CM version 2, the CM-contact *Security Required* parameter as the D-START *Security Requirements* parameter value,

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

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

d) enter the *CONTACT* state.

2.5.3.3.8.2 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,

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

d) enter the *CONTACT DIALOGUE* state.

2.5.3.3.9 CM-end Service Request

2.5.3.3.9.1 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.

2.5.3.3.10 CM-forward Service Request

2.5.3.3.10.1 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 or, if CM version 2 and the *Emulated Version* parameter is not provided, a cmEnhancedForwardRequest 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) for CM version 2, if the *Emulated Version* parameter value is not provided, the sending CM-ground-ASE version number as the D-START *DS User Version Number* parameter value,

4) for CM version 2, if the *Emulated Version* parameter is provided, the Emulated Version parameter value as the D-START *DS User Version Number* parameter value,

5) if CM version 2 and the *Security Required* parameter is provided, the CM-forward *Security Required* parameter value as the D-START *Security Requirements* parameter value, and

6) 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,

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

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

d) enter the *FORWARD* state.

2.5.3.3.11 CM-user-abort Service Request

2.5.3.3.11.1 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.

2.5.3.3.12 D-ABORT Indication

2.5.3.3.12.1 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.

2.5.3.3.13 D-P-ABORT Indication

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

c) a) stop any timer, if set,

d) 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

e) enter the *IDLE* state.

2.5.4 Exception Handling

2.5.4.1 Timer Expiration

2.5.4.1.1 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.

2.5.4.2 Unrecoverable System Error

2.5.4.2.1 **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.*

2.5.4.3 Invalid PDU

2.5.4.3.1 If the *User Data* parameter of a D-START indication or D-DATA indication does not contain a valid PDU as defined in 2.5.3.1.3 and 2.5.3.1.4, 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.

2.5.4.3.2 If the *User Data* parameter of a D-START confirmation does not contain a valid PDU as defined in 2.5.3.1.3 and 2.5.3.1.4, the CM-ASE shall:

a) stop all timers,

b) if the CM-ASE is a CM-air-ASE and if the D-START *Result* parameter is set to the abstract value "accepted", then

1) create a CMAircraftMessage APDU with a cmAbortReason [invalid-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,

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.

2.5.4.4 Not Permitted PDU or Dialogue Service Primitive

2.5.4.4.1 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 2.5.3.1.2, 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 [protocol-error] APDU message element,

c) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [protocol-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 "protocol-error" as the CM-provider-abort *Reason* parameter value, and

f) enter the *IDLE* state.

2.5.4.4.2 If the *User Data* parameter of a D-START confirmation is a valid PDU, but is not a permitted PDU as defined within 2..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 [protocol-error] APDU message element,

2) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [protocol-error] 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 "protocol-error" as the CM-provider-abort *Reason* parameter value, and

d) enter the *IDLE* state.

2.5.4.4.3 Upon receipt of a Dialogue service primitive for which there are no instructions in 2.5.3 (i.e. the primitive was not expected or was expected under other conditions or with other parameter values), 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 [protocol-error] APDU message element,

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

d) if a dialogue exists, 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 "protocol-error" as the CM-provider-abort *Reason* parameter value, and

f) enter the *IDLE* state.

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

2.5.4.5.1 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.

2.5.4.5.2 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", the CM-ASE shall:

a) stop all timers, and

b) if the CM-user is an active user, invoke CM-provider-abort service indication with the abstract value "communication-service-error" APDU as the CM-provider-abort *Reason* parameter value.

2.5.4.6 D-END Confirmation Not as Expected

2.5.4.6.1 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.

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

2.5.4.7.1 If the abstract value of the D-START *Priority* QOS parameter is not "flight regularity communications" or the abstract value of the D-START *RER* QOS parameter is not "low" or the abstract value of the D-START *Routing Class* QOS parameter does not identify the traffic category "Air Traffic Service Communications (ATSC)", 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

2.5.4.8 Expected PDU Not Provided

2.5.4.8.1 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.

2.5.4.8.2 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.

2.5.4.9 D-START Security Requirements Parameter Not as Expected

Note.— This section applies to CM version 2 only. This is for the case when the D-START Security Requirements *parameter does not meet the local security policy.*

2.5.4.9.1 Upon receipt of a D-START indication with the *Security Requirements* parameter not consistent with the local security policy or upon receipt of a D-START confirmation with the *Security Requirements* parameter not equal to the value that was set in the D-START request, the CM-ASE shall:

a) stop all timers,

b) if the dialogue with the peer is still open:

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

2) if the CM-ASE is a CM-ground-ASE, create a CMGroundMessage APDU with a cmAbortReason [communication-service-failure] 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 active, invoke a CM-provider-abort service indication with the abstract value "communication-service-failure", and

d) enter the *IDLE* state.

2.5.5 CM ASE State Tables

2.5.5.1 Priority

2.5.5.1.1 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 2.5-2. CM-Ground-ASE State Table

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
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	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	cannot occur	cannot occur
D-START Indication, *User Data* = CMServerFacilityQuery Request (version 2)	●CM-server-facility-query indication →*SERVER QUERY*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
D-START Indication *Version Number* is less than or equal to the CM-ground-ASE version number, *User Data* = CMForwardRequest or CMEnhancedForwardRequest, ground-ground forwarding supported (version 2)	●CM-forward indication ●D-START response →*IDLE*	cannot occur	cannot occur	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 or CMEnhancedForwardRequest (version 2)	●D-START response →*IDLE*	cannot occur	cannot occur	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	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	cannot occur	cannot occur

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user", D-START User *Data*=CMForwardResponse or CMEnhancedForwardRequest (version 2)	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●Stop timer $t_{forward}$ ●CM- forward confirmation →*IDLE*	cannot occur	cannot occur
D-DATA Indication, D-DATA *User Data* = CMContactResponse	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●CM contact confirmation ●Stop timer $t_{contact}$ →*DIALOGUE*	cannot occur	cannot occur	cannot occur	cannot occur
D-DATA Indication, D-DATA *User Data* = CMServerFacilityQuery Request (version 2)	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●Discard D-DATA indication →*CONTACT DIALOGUE*	●Discard D-DATA indication →*END*	cannot occur	cannot occur	●CM server facility query indication →*DIALOGUE*
D-END Confirmation *Result* "accepted"	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●stop timer t_{end} →*IDLE*	cannot occur	cannot occur	cannot occur
CM-User Events										

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
CM-update Request or CM-server-facility-update Request (version 2)	●D-START request ●Start timer t_{update} →*UPDATE*	not permitted	not permitted	not permitted	●D-DATA request →*DIALOGUE*	not permitted	not permitted	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 ●Stop timer $t_{contact}$ →*CONTACT DIALOGUE*	not permitted	not permitted	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	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	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	not permitted	not permitted
CM-server-facility-query Response *Maintain Dialogue* not supplied by CM-ground user (version 2)	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	●D-START response →*IDLE*	not permitted

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
CM-server-facility-query Response *Maintain Dialogue* "accepted" (version 2)	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted	●D-START response →*DIALOGUE*	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	not permitted	not permitted
ABORT Events										
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 ●Stop timer $t_{contact}$ →*IDLE*	not permitted	●stop timer $t_{forward}$ ●D-ABORT request →*IDLE*	●stop timer t_{srv_query} ●D-ABORT request →*IDLE*	●stop timer t_{srv_query} ●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 ●Stop timer $t_{contact}$ →*IDLE*	●stop timer t_{end} →*IDLE*	●stop timer $t_{forward}$ ●CM-provider-abort indication →*IDLE*	●stop timer t_{srv_query} ●CM-provider-abort indication →*IDLE*	●stop timer t_{srv_query} ●CM-provider-abort indication →*IDLE*

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
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 ●Stop timer $t_{contact}$ →*IDLE*	●stop timer t_{end} ⇒*IDLE*	cannot occur	●stop timer t_{srv_query} ●CM-user-abort indication →*IDLE*	●stop timer t_{srv_query} ●CM-user-abort indication →*IDLE*
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 ●Stop timer $t_{contact}$ →*IDLE*	●stop timer t_{end} →*IDLE*	●stop timer $t_{forward}$ ●CM- provider- abort indication →*IDLE*	●stop timer t_{srv_query} ●CM- provider- abort indication →*IDLE*	●stop timer t_{srv_query} ●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	cannot occur	cannot occur

STATE → EVENT ⇓	*IDLE*	*LOGON*	*UPDATE*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*END*	*FORWARD*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE* *(version 2)*
$T_{contact}$ Expires	cannot occur	cannot occur	cannot occur	●D-ABORT request ●CM-provider- abort indication →*IDLE*	cannot occur	●D-ABORT request ●CM-provider- abort indication →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur
T_{end} Expires	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●D-ABORT request →*IDLE*	cannot occur	cannot occur	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*	cannot occur	cannot occur

Table 2.5-3. CM-Air-ASE State Table

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE (version 2)*
DIALOGUE Service Events							
D-START Indication *User Data* CMUpdate or CMSecureUpdate (version 2)	●CM-update indication ●D-START response →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication *User Data* CMServerFacilityUpdate (version 2)	●CM-server-facility-update indication ●D-START response →*IDLE*	cannot occur	cannot occur	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	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	cannot occur	cannot occur

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE (version 2)*
D-START Confirmation *Result* "accepted" *User Data* CMLogonResponse or CMSecureLogonResponse (version 2)	cannot occur	●Stop timer t_{logon} ●CM-logon confirmation →*DIALOGUE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user" *User Data* CMServerFacilityQuery Response (version 2)	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●Stop timer t_{srv_query} ●CM-logon confirmation →*IDLE*	cannot occur
D-START Confirmation *Result* "accepted" *User Data* CMServerFacilityQuery Response (version 2)	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●Stop timer t_{srv_query} ●CM-logon confirmation →*DIALOGUE*	cannot occur
D-DATA Indication *User Data* CMUpdate	cannot occur	cannot occur	cannot occur	●CM-update indication →*DIALOGUE*	cannot occur	cannot occur	●Discard D-DATA indication →*SERVER QUERY DIALOGUE*

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE (version 2)*
D-DATA Indication *User Data* CMContactRequest	cannot occur	cannot occur	cannot occur	●CM-contact indication *→CONTACT DIALOGUE*	cannot occur	cannot occur	●stop timer t_{srv_query} ●CM-server-facility-query confirmation ●CM-contact indication *→CONTACT DIALOGUE*
D-DATA Indication *User Data* CMServerFacilityQuery Response (version 2)	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●stop timer t_{srv_query} ●CM-server-facility-query confirmation *→ DIALOGUE*
D-DATA Indication *User Data* CMServerFacilityUpdate (version 2)	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●Discard D-DATA indication *→SERVER QUERY DIALOGUE*

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE (version 2)*
D-END Indication	cannot occur	cannot occur	cannot occur	●CM-end indication ●D-END response *→IDLE*	cannot occur	cannot occur	●stop timer t_{srv_query} ●CM-server-facility-query confirmation ●CM-end indication ●D-END response *→IDLE*
CM-User Events							
CM-contact Response	not permitted	not permitted	●D-START response *→IDLE*	not permitted	●D-DATA request *→DIALOGUE*	not permitted	not permitted
CM-logon Request	●D-START request ●Start timer t_{logon} *→LOGON*	not permitted	not permitted	not permitted	not permitted	not permitted	not permitted

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE (version 2)*
CM-server-facility-query Request (version 2)	●D-START request ●Start timer t_{srv_query} →*SERVER QUERY*	not permitted	not permitted	●D-DATA request ●Start timer t_{srv_query} →*SERVER QUERY DIALOGUE*	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*	●stop timer t_{srv_query} ●D-ABORT request →*IDLE*	●stop timer t_{srv_query} ●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*	●stop timer t_{srv_query} ●CM-provider-abort indication →*IDLE*	●stop timer t_{srv_query} ●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*	●stop timer t_{srv_query} ●CM-user-abort indication →*IDLE*	●stop timer t_{srv_query} ●CM-user-abort indication →*IDLE*

STATE → EVENT ⇓	*IDLE*	*LOGON*	*CONTACT*	*DIALOGUE*	*CONTACT DIALOGUE*	*SERVER QUERY* *(version 2)*	*SERVER QUERY DIALOGUE (version 2)*
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*	●stop timer t_{srv_query} ●CM-provider-abort indication →*IDLE*	●stop timer t_{srv_query} ●CM-provider-abort indication →*IDLE*
T_{logon} Expires	cannot occur	●D-ABORT request ●CM-provider-abort indication →*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
T_{srv_query} Expires	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur	●D-ABORT request ●CM-provider-abort indication →*IDLE*	●D-ABORT request ●CM-provider-abort indication →*IDLE*

2.6 COMMUNICATION REQUIREMENTS

2.6.1 Encoding Rules

2.6.1.1 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 2.4.

Note.— When encoded CM APDUs are treated as bit-oriented values that are not padded to an integral number of octets, the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

2.6.2 Dialogue Service Requirements

2.6.2.1 Primitive Requirements

2.6.2.1.1 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..5, the CM-ground-ASE and the CM-air-ASE shall exhibit external behaviour consistent with the dialogue service, as described in Doc 9705, Sub-volume IV, paragraph 4.2, having been implemented and its primitives invoked.

2.6.2.2 ATN Quality-of-Service Requirements

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

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

2.6.2.2.3 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 2.6-1.

Table 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.— ATSCclass values are defined in Doc 9705, Sub-volume 1, section 1-3 (Table 1-1).

2.6.2.3 ATN Security Requirements

Note.— CM version 1 does not use this parameter.

2.6.2.3.1 For CM version 2, the Security Requirements parameter of the D-START shall be set to either “Secured Dialogue supporting key management” or “No security” for the CM-logon service.

2.6.2.3.2 For CM version 2, the Security Requirements parameter of the D-START shall be set to either “Secured Dialogue supporting key management”, “Secured Dialogue” or “No security” for the CM-update service, CM-server-facility-update service and CM-server-facility-query service.

Note.— The value “Secured Dialogue supporting key management” is used if there has not been a previous CM-logon performed with the peer CM application. This is in order to establish the session key (see Doc 9705, Sub-volume VIII for more details). The value “Secured Dialogue” is used if there has been a previous CM-logon performed with the peer CM application.

2.6.2.3.3 For CM version 2, the Security Requirements parameter of the D-START shall be set to either "Secured Dialogue" or "No security" for the CM-contact service.

2.6.2.3.4 For CM version 2, the Security Requirements parameter of the D-START shall be set to either “Secured Dialogue” or “No security” for the CM-forward service.

2.7 CM USER REQUIREMENTS

Note.— Requirements imposed on CM-users concerning CM messages and interfacing with the CM-ASEs are presented in 2.7.

2.7.1 CM-Air-User Requirements

2.7.1.1 General CM-air-user Requirements

2.7.1.1.1 A version 2 CM-air-user shall be prohibited from invoking any CM version 2-specific service or using any CM version 2 parameters (other than the Security Required parameter) with a version 1 CM ground system.

Note 1.— When a CM-air-user invokes the CM-logon or, if CM version 2, the CM-server-facility-query 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 or, if CM version 2, the CM-server-facility-query 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 the CM-air-user invokes the CM-logon or, if CM version 2, the CM-server-facility-query service for version 2, it sets the Security Required *parameter as appropriate to the local security policy.*

Note 4.— For each CM-air-ASE invocation, the CM-air-user establishes a correlation between a CM-air-ASE invocation and the facility designation.

Note 5.— Upon the initiation of a CM-logon or, if CM version 2, the CM-server-facility-query service request, or upon receipt of a CM-update or, if CM version 2, the CM-server-facility-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 6.— The correlation is maintained for the duration of the ASE invocation.

2.7.1.1.1.1 When communicating with a version 1 CM ground system, the CM-air-user shall be prohibited from:

a) sending a CM-server-facility-query request; and

b) using the *Security Required* parameter with any value other than “No security”.

2.7.1.2 CM-logon Service Requirements

Note 1.— Only the CM-air-user is permitted to initiate the CM-logon service.

Note 2.— For version 2 CM, either secure or unsecure CM-logon services may be performed.

2.7.1.2.1 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,

d) the facility designation of the facility with which the CM-air-user wishes to exchange application information (if required), and

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

Note.— The facility designation *is only used if the CM-air-user wants to explicitly identify a facility other than the one contained in the* Facility Designation *parameter of the CM-logon service for which it requires application information.*

2.7.1.2.2 **Recommendation.—** *The CM-air-user should only use the facilityDesignation field of the CMLogonRequest if requesting information for a facility other than the one specified in the Facility Designation parameter of the CM-logon service.*

2.7.1.2.3 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.

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

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

2.7.1.2.5 For CM version 2, when invoking a CM-logon request, if the CM-air-user requires to use a previous CM-air-ASE version, the CM-air-user shall set the *Emulated Version* parameter value to the required version number.

Note.— This may be done for a variety of reasons, including a priori knowledge of a lower version CM ground system.

2.7.1.2.6 For CM version 2, when the CM-air-user provides the *Emulated Version* parameter, the *Emulated Version* parameter value shall be less than the actual CM-air-ASE version number.

Note.— This is to ensure that services not supported by a peer CM ground system will not be invoked.

2.7.1.2.7 For CM version 2, if the CM-air-ASE is operating in a lower version emulation mode, the CM-air-user shall only invoke or accept CM services supported by that *CM ASE Version Number*.

Note.— This means that the CM-air-user must not invoke any services or use any parameters not known to the lower-versioned peer CM ground system.

2.7.1.2.8 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 2.4.

Note 1.— The actual TSAP = IDP + long TSAP. The IDP = AFI + IDI.

Note 2.— If the Version Number *parameter is not provided and the* Logon Response *parameter is provided but is empty, this means that the CM-ground-user has rejected the logon. This may be for operational reasons (e.g. optional information that is required for a particular airspace was missing in the CM-logon request) or technical reasons (e.g. there is a problem with the ground system that prevents it from providing the information). This case will NOT be due to a version number incompatibility, since in that case only the* Version Number *is provided. If a CM-logon service is rejected, the CM-air-user may evaluate possible reasons for the rejection and retry the CM-logon service.*

2.7.1.2.9 Upon receipt of a CM-logon service confirmation, the CM-air-user shall make the information contained in the CMLogonResponse or, if CM version 2, the CMSecureLogonResponse available to the other applications (i.e. ADS, CPDLC, and FIS), as well as to the dialogue service provider.

Note.— In the case of the CMSecureLogonResponse, the information to be made available to other applications includes key information if provided. Only a version 2 CM will receive a CMSecureLogonResponse.

2.7.1.2.10 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.

Note 1.— If the facility designation field of the Logon Request was provided, then the information contained in the Logon Response parameter corresponds to that facility designation. If the facilityDesignation field of the LogonRequest was not provided, then the information contained in the Logon Response parameter corresponds to the Facility Designation parameter of the CM-logon service.

Note 2.— If the facility designation field of the Logon Request was provided and no application information is returned in the Logon Response parameter, this means that the CM-ground-user does not have access to the information for the requested facility or that the requested facility does not support any of the proposed applications.

2.7.1.2.11 **Recommendation.—** *For CM version 2, upon receipt of a CM-logon service confirmation with the CM ASE Version Number provided, the CM-air-user should invoke a CM-logon service request using the Emulated Version parameter as defined in 2.7.1.2.5.*

Note.— If the CM ASE Version Number is provided in the CM-logon service confirmation, it means that the CM-air-ASE version number is greater than the CM-ground-ASE version number. In order for the CM-air-ASE to perform a successful CM-logon service with that lower version CM ground system, the Emulated Version parameter must be used. This option is only available to CM version 2.

2.7.1.3 CM-update Service Requirements

Note.— If a CM-update service indication is received, then the information contained in the Update Information parameter corresponds to the facility designation contained in the Facility Designation parameter, or, if a dialogue exists, the facility with which the dialogue is in place.

2.7.1.3.1 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.

2.7.1.3.2 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 2.4.

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

2.7.1.3.3 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.

Note.— In the case of the CMSecureUpdate for CM version 2, the information to be made available to other applications includes key information, if provided.

2.7.1.4 CM-contact Service Requirements

2.7.1.4.1 **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.*

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

2.7.1.4.3 **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.*

2.7.1.4.4 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 2.7.1.1. However, the facility designation will not be provided as part of the CMLogonRequest in this case.

2.7.1.4.5 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.

2.7.1.5 CM-server-facility-query Service Requirements

Note 1.— Only the CM-air-user is permitted to initiate the CM-server-facility-query service.

*Note 2.— Either secure or unsecure CM-server-facility-query services may be performed. This service is only avail*able to version 2 CM users.

2.7.1.5.1 When invoking the CM-server-facility-query request the CM-air-user shall provide the following as part of the CMServerFacilityQueryRequest:

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) the facility designations of up to eight facilities that the aircraft desires application information from, and

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

2.7.1.5.2 When invoking the CM-server-facility-query 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.

2.7.1.5.3 When invoking the CM-server-facility-query service request, the ARS component of the short TSAP shall contain the ICAO 24 bit aircraft address.

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

2.7.1.5.4 Upon receipt of a CM-server-facility-query service confirmation, the CM-air-user shall create the actual TSAP for each ground application information contained in the *Server Facility Query Response* based on the IDP and long TSAP for each application as defined in 2.4.

Note 1.— The actual TSAP = IDP + long TSAP. The IDP = AFI + IDI.

Note 2.— The CM-air-user establishes a correlation between the addresses received in the CM-server-facility-query service confirmation and the facilities to which they belong. The CM-air-user may then perform data link services with the applications at those facilities.

2.7.1.5.5 Upon receipt of a CM-server-facility-query service confirmation, the CM-air-user shall make the information for each facility contained in the *Server Facility Query Response* available to the other applications (i.e. ADS, CPDLC, and FIS), as well as to the dialogue service provider.

Note.— The information to be made available to other applications includes key information if provided.

2.7.1.5.6 Upon the receipt of a *Server Facility Query Response* which contains information for 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.

2.7.1.6 CM-server-facility-update Service Requirements

Note.— This service is only available to CM version 2.

2.7.1.6.1 Upon the receipt of a *Server Facility Update Information* which contains information for a ground facility 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.

2.7.1.6.2 Upon receipt of a CM-server-facility-update service indication, the CM-air-user shall create the actual TSAP for each ground application information contained in the *Server Facility Update Information* based on the IDP and long TSAP for each application as defined in 2.4.

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

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

Note.— The information to be made available to other applications includes key information, if provided

2.7.1.7 CM-user-abort Service Requirements

Note 1.— When an CM-air-user requires to abort the current service or maintained dialogue, it initiates a CM-user-abort request.

Note 2.— A CM-air-user may require to abort the current service or maintained dialogue for several reasons, including the detection of an unrecoverable error situation and the local policies. The user abort request and indication primitives do not indicate the reason of the abort.

2.7.2 CM-Ground-User Requirements

2.7.2.1 General CM-Ground-User Requirements.

2.7.2.1.1 A CM-ground-user shall invoke the CM-logon service, CM-update service, CM-contact service, CM-end service and, if CM version 2, the CM-server-facility-query service and CM-server-facility-update service only when communicating with a CM-air-user.

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

2.7.2.1.3 A version 2 CM-ground-user shall be prohibited from invoking any CM version 2-specific service or using any CM version 2 parameters (other than the *Security Required* parameter) with a version 1 CM ground or air system.

2.7.2.1.4 When communicating with a version 1 CM aircraft, the CM-ground-user shall be prohibited from:

a) sending a CM-server-facility-query response;

b) sending a CM-server-facility-update request;

c) sending a CM-logon response with a CMSecureLogonResponse parameter;

d) sending a CM-update request with a CMSecureUpdate parameter; and

e) using the Security Required parameter with any value other than “No security”.

Note.— In some cases the Security Required parameter may not be provided.

2.7.2.1.4.1 When communicating with a version 1 CM ground system, the CM-ground-user shall be prohibited from:

a) sending a CM-forward request with a *CMEnhancedForwardRequest* parameter; and

b) using the *Security Required* parameter with any value other than "No security".

2.7.2.1.4 For version 2 services requiring key information (CM-logon, CM-server-facility-query, CM-update and CM-server-facility-update), the CM-ground-user shall obtain the authentic copy of the key agreement public key and domain usage information for each application that requires security.

Note.— The certification of the keys themselves is handled by the secure Dialogue Service Provider, as described in Doc 9705, Sub-volume IV, paragraph 4.8. However, the CM-user is responsible for validating the key agreement public key certificates. The certification of the keys themselves may be handled through direct invocation of SSO services.

2.7.2.1.5 **Recommendation.—** *For CM version 2, the CM-ground-user should use the ATN Directory service as defined in Doc 9705, Sub-volume VII to obtain key agreement public key and domain usage information.*

Note 1.— Directory implementation and policy is local implementation.

Note 2.— The CM-ground-user may choose to update aircraft information held in a central directory. This may be done by use of the ATN Directory Service as defined in Doc 9705, Sub-volume VII.

Note 3.— When a CM-ground-user invokes the CM-update service, CM-contact service, CM-forward service, or, if CM version 2, the CM-server-facility-update 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 4.— When the CM-ground-user invokes a CM-update service, CM-contact service, CM-forward service, or, if CM version 2, the CM-server-facility-update 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 5.— 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 6.— When the CM-ground-user invokes the CM-update, CM-server-facility-update, CM-contact or CM-forward request service for version 2, it sets the Security Required parameter as appropriate to the local security policy.

Note 7.— For each CM-ground-ASE invocation, the CM-ground-user establishes a correlation between a CM-ground-ASE invocation and the ICAO t 24 bit aircraft address.

Note 8.— Upon the initiation a CM-update or, if CM version 2, CM-server-facility-update service request or CM-contact service request, or upon receipt of a CM-logon or, if CM version 2, CM-server-facility-

query service indication the ASE invocation correlation is based on the ICAO 24 bit aircraft address in the Aircraft Address parameter of the respective CM service.

Note 9.— The correlation is maintained for the duration of the ASE invocation

2.7.2.2 CM-logon Service Requirements

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

2.7.2.2.2 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.

2.7.2.2.3 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 2.4.

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

2.7.2.2.4 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.

2.7.2.2.5 Upon receipt of a CM-logon service indication from a CM version 1 peer, 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.

Note.— This is the case for a version 1 CM aircraft performing the CM-logon service. The CM-ground-user can determine the version of the peer by an indication of the Version Number *parameter in the CM-logon indication if the CM-air-ASE version is lower than the CM-ground-ASE version (as is the case for a CM version 2 CM-ground-ASE communicating with a version 1 CM-air-ASE), or by the absence of the* Version Number *parameter in the CM-logon indication if the CM-air-ASE version is the same as the CM-ground-ASE version (as is the case for a version 1 CM-ground-ASE communicating with a version 1 CM-air-ASE or a version 2 CM-air-ASE in emulation mode).*

2.7.2.2.6 For CM version 2, upon receipt of a CM-logon service indication with the *Security Required* parameter provided with the value “Secured Dialogue supporting key management”, the CM-ground-user shall invoke a CM-logon service response with a CMSecureLogonResponse containing:

a) the facility designation of the facility to which the information applies,

Note.— The facility designation is only provided if the information is for a facility other than the responding facility.

b) application names, addresses, version numbers, and optionally, authentic key agreement public keys and domain usage indications for the requested applications that can be air-initiated for all versions that the ground and aircraft systems can support, and

c) application names, version numbers, and optionally, authentic key agreement public keys and domain usage indications for the requested ground-only initiated applications that the ground system can support.

2.7.2.2.7 For CM Version 2, upon receipt of a CM-logon service indication with the *Security Required* parameter provided with the value "No security" and when communicating with a version 1 CM peer, the CM-ground-user shall invoke a CM-logon service response with a CMSecureLogonResponse containing:

a) the facility designation of the facility to which the information applies,

Note.— The facility designation is only provided if the information is for a facility other than the responding facility.

b) 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

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

Note.— A CM-air-user will not provide security information if an unsecure dialogue is being maintained.

2.7.2.2.8 When communicating with a CM version 1 peer, if the facility designation is present in the *Logon Request* parameter, then the application information contained in the *Logon Response* shall correspond to that facility designation.

Note 1.— The information for the requested facility may be obtained through the use of the ATN Directory Service (See Doc 9705, Sub-volume VII).

Note 2.— If a CM-ground-user does not have access to the information for the requested facility, no application information is returned.

2.7.2.2.9 When communicating with a CM version 1 peer, if the facility designation is not present in the *Logon Request* parameter, then the application information contained in the *Logon Response* shall correspond to the responding CM-ground-user's facility designation.

2.7.2.2.10 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.

2.7.2.2.11 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.

2.7.2.2.12 If a CM-ground-user wishes to reject a CM-logon service indication for any reason, the CM-ground-user shall invoke a CM-logon service response with a *Logon Response* containing no information.

Note 1.— This may be done for either operational reasons (e.g. optional information that is required for a particular airspace is missing or information for the requested facility is not available) or technical reasons other than version incompatibility (e.g. there is a problem with the ground system and the service is not available). These example reasons are not meant to be an exhaustive list.

Note 2.— This will not be done if the CM-air-ASE and CM-ground-ASE versions are incompatible. In that case, the CM-ground-ASE will reject the D-START indication as detailed in 2.5 before it reaches the CM-ground-user.

2.7.2.3 CM-update Service Requirements

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

Note 2.— For version 2 CM, either secured or unsecured CM-update service may be performed.

2.7.2.3.1 When invoking the CM-update service request with version 1 CM aircraft, the CM-ground-user shall provide a CMUpdate containing application names, addresses, and version numbers for each of the data link applications being updated.

Note 1.— This applies to both CM version 1 and CM version 2.

Note 2.— The CM-update service only corresponds to a ground facility's local applications.

2.7.2.3.2 For CM version 2, when invoking the CM-update service request with version 2 CM aircraft, if

use of security is requested, the CM-ground-user shall provide a CMSecureUpdate containing:

a) the facility designation of the facility to which the information applies,

Note.— The facility designation is only provided if the information is for a facility other than the responding facility.

b) application names, addresses, version numbers, and optionally, authentic key agreement public keys and domain usage indications for the requested applications that can be air-initiated for all versions that the ground and aircraft systems can support, and

c) application names, version numbers, and optionally, authentic key agreement public keys and domain usage indications for the requested ground-only initiated applications that the ground system can support

2.7.2.3.3 For CM version 2, when invoking the CM-update service request with version 2 CM aircraft, if use of security is not requested, the CM-ground-user shall provide a CMSecureUpdate containing:

a) the facility designation of the facility to which the information applies,

Note.— The facility designation is only provided if the information is for a facility other than the responding facility.

b) 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

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

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

2.7.2.4 CM-contact Service Requirements

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

Note 2.— For version 2 CM, either secure or unsecure CM-contact services may be performed.

2.7.2.4.1 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.

2.7.2.5 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.*

2.7.2.6 CM-forward Service Requirements

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

Note 2.— For version 2 CM, either secure or unsecure CM-forward service may be performed.

2.7.2.6.1 When requesting the CM-forward service for version 1 CM, the CM-ground-user shall provide a CMForwardRequest containing all of the information from either a CM-logon request message or a CM-forward request message, whichever is the more recent.

Note.— This applies to both CM version 1 and CM version 2 in emulation mode.

2.7.2.6.2 For CM version 2, when requesting the CM-forward service for version 2 CM, the CM-ground-user shall provide a CMEnhancedForwardRequest containing all of the information from either a CM-logon request message or a CM-forward request message, whichever is the more recent, and an indication of the aircraft's CM version number applicable to the forwarded information.

2.7.2.6.3 For CM version 2, when invoking a CM-forward request, if the CM-ground-user requires to use a previous CM-ground-ASE version, the CM-ground-user shall set the *Emulated Version* parameter value to the required version number.

Note.— This may be done for a variety of reasons, including a priori knowledge of a lower version CM ground system.

2.7.2.6.4 **Recommendation.—** *When a CM version 2 performs an emulated CM version 1 CM-forward request, the initiating CM-ground-user should provide the CM version 2 information in the CMForwardRequest.*

Note.— This is done in case the CM version 1 ground system subsequently performs a CM-forward with a CM version 2 ground system. The receiving CM version 2 ground system will then be able to take full advantage of the aircraft's capabilities.

2.7.2.6.5 For CM version 2, when the CM-ground-user provides the *Emulated Version* parameter, the *Emulated Version* parameter value shall be less than the actual CM-ground-ASE version number.

Note.— This is to ensure that parameters not supported by a peer CM ground system will not be used.

2.7.2.6.6 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.

2.7.2.6.7 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..4.

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

2.7.2.6.8 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.

2.7.2.6.9 **Recommendation.—** *Upon the receipt of a* Forward Request *from a CM-forward service indication, the receiving CM-ground-user should invoke a CM-update service request with the indicated aircraft, if the update service is supported.*

2.7.2.7 CM-server-facility-query Service Requirements

Note.— This service applies to CM version 2 only.

2.7.2.7.1 Upon receipt of a CM-server-facility-query service indication, the CM-ground-user shall make the aircraft application information contained in the *Server Facility Query Request* available to the other applications (i.e. ADS, CPDLC, and FIS), as well as to the dialogue service provider.

2.7.2.7.2 Upon receipt of a CM-server-facility-query service indication, the CM-ground-user shall create the actual TSAP for each aircraft application information contained in the *Server Facility Query Request* based on the IDP and long TSAP for each application as defined in 2.4.

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

2.7.2.7.3 Upon the receipt of a *Server Facility Query Request* from a CM-server-facility-query 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.

2.7.2.7.4 Upon receipt of a CM-server-facility-query service indication, if the CM-ground-user does not support the ability to retrieve other facilities' application information or if the ability to retrieve other facilities' application information is temporarily unavailable, the CM-ground-user shall invoke a CM-server-facility-query service response indicating the reason.

Note.— This is done with via the "InfoUnavailable" ASN.1 choice. In this case, either non-support of the server may be indicated (the "serverNotSupported" option) or temporary unavailability of server access may be indicated (the "serverUnavailable" option). The "serviceInterrupted" option is only used by the CM-air-ASE during a collision condition.

2.7.2.7.5 Upon receipt of a CM-server-facility-query service indication, the CM-ground-user shall

invoke a CM-server-facility-query service response with a *Server Facility Query Response* containing application information for each facility designation, if available, as follows:

a) the facility designation for which each set of application information is relevant,

b) 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,

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

d) if the CM-server-facility-query service indication *Security Required* parameter contained the abstract value "Secured Dialogue supporting key management" or "Secured Dialogue" (i.e. if secure services are requested and supported), key and domain usage information for each application that requires security.

Note.— If information for a particular facility is not available, then no application information for that facility will be returned to the aircraft.

2.7.2.7.6 **Recommendation.—** *For the CM-server-facility-query service, the CM-ground-user should use the ATN Directory service to obtain application information relating to other facilities.*

2.7.2.7.7 **Recommendation.**— *The CM-ground-user should only return information for the facility designations specified in the* Server Facility Query Request*, if available.*

2.7.2.7.8 When invoking the CM-server-facility-query service response, if any RDP for a given facility's 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.

2.7.2.7.9 When the CM-ground-user requires a CM dialogue to be maintained, the CM-ground-user shall set the CM-server-facility-query service response Maintain Dialogue parameter, if and only if the dialogue maintain service is supported.

2.7.2.8 CM-server-facility-update Service Requirements

Note.— Only the CM-ground-user is permitted to initiate the CM-server-facility-update service. This service applies to CM version 2 only.

2.7.2.8.1 When invoking the CM-server-facility-update service request, the CM-ground-user shall provide a *Server Facility Update Information* containing:

a) the facility designation for which each set of application information is relevant,

b) 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,

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

d) if secure services are requested and supported, key and domain usage information for each application that requires security.

2.7.2.8.2 **Recommendation.—** For *the CM-server-facility-update service, the CM-ground-user should use the ATN Directory service to obtain application information relating to other facilities.*

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

2.7.2.9 CM-user-abort Service Requirements

Note 1.— When an CM-ground-user requires to abort the current service or maintained dialogue, it initiates a CM-user-abort request.

Note 2.— A CM-ground-user may require to abort the current service or maintained dialogue for several reasons, including the detection of an unrecoverable error situation and the local policies. The user abort request and indication primitives do not indicate the reason of the abort.

2.7.3 Parameter Value Unit, Range and Resolution

2.7.3.1 A CM user shall interpret parameter value unit, range and resolution as defined in 2.4.

2.8 SUBSETTING RULES

2.8.1 General

Note.— 2.8.1.1 specifies conformance requirements to which all implementations of the CM protocol obey.

2.8.1.1 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 tables 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 2.8-1. CM Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	C	V1
Version 2	C	V2

C: a conformant implementation shall support one and only one of these two options

Note 2.— A version 2 CM inherently has the capability to emulate a version 1 CM.

Table 2.8-2. CM Operational Functional Units

	Status	**Associated Predicate**
The CM system acts as an airborne system	C.1	CM/air
The CM system acts as a ground system	C.1	CM/ground
The ground CM system can update application information	if (CM/ground) O, else N/A	G-UP-FU
The ground CM system can request the aircraft to initiate a logon with a specified CM ground system	if (CM/ground) O, else N/A	G-CO-FU
The ground CM user can process forwarded aircraft information	if (CM/ground) O, else N/A	G-FO-FU
The ground CM system can forward aircraft information to another CM ground system	if (CM/ground) O, else N/A	G-FO-IN
The ground CM user can initiate a server facility update with a specified air system	if (CM/ground) O, else N/A	G-SF-FU
The ground CM user can process a server facility query	if (CM/ground) O, else N/A	G-SQ-FU
The air CM user can process an update request	if (CM/air) O, else N/A	A-UP-FU
The air CM user can process an contact request	if (CM/air) O, else N/A	A-CO-FU
The air CM user can initiate a server facility query with a specified ground system	if (CM/air) O, else N/A	A-SF-FU
The air CM user can process a server facility update	if (CM/air) O, else N/A	A-SU-FU

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

Note 3.— A conformant CM-air implementation will consist of at least the core functionality (CM/air) with, optionally, any combination of the A-UP-FU, A-CO-FU, A-SF-FU, and A-SU-FU functional units. If only the core functionality is supported, then logon exchanges can be initiated with ground systems, received contact requests are rejected with reason 'contact not successful', received update and server-facility-update requests are processed by the CM-air-ASE but ignored by the CM-air-user, and the capability to maintain dialogue is supported. Mandatory support for security is inherent in any CM version 2 implementation.

Note 4.— A conformant CM-ground implementation will consists of at least the core functionality (CM/ground) with, optionally, any combination of the G-FO-IN, G-FO-FU, G-CO-FU, G-UP-FU, G-SF-FU, and G-SQ-FU. If only the core functionality is supported, then logon exchanges are supported with an aircraft, received forward requests are rejected with reason 'service not supported', received server facility queries are rejected with 'server not supported', and the capability to maintain dialogue optionally supported. Mandatory support for security is inherent in any CM version 2 implementation.

Note 5.— The Protocol/Operational Implementation Conformance Statements (P/OICS) templates provide the capability to detail all implementation details to a much deeper level than can be given by the functional unit descriptions of 2.8. All details for both air and ground implementations can be found in completed P/OICS for particular States' or industries' implementations.

— — — — — — — —

Page left intentionally
blank

Chapter 3 — Controller-pilot data link communications

(Sections 3.1, 3.2 and 3.3)

(See mapping table for conversion of current paragraph numbers of Doc 9705 – 3rd edition into paragraph numbers of Doc 9880)

3. CONTROLLER PILOT DATA LINK COMMUNICATION APPLICATION

3.1 General

3.1.1 This chapter contains the following paragraphs:

3.2 GENERAL REQUIREMENTS contains the CPDLC version number, and error processing requirements.

3.3 ABSTRACT SERVICE DEFINITION contains the description of the abstract service provided by the CPDLC Application Service Element (CPDLC-ASE).

3.4 FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by CPDLC ASEs using Abstract Syntax Notation Number One (ASN.1).

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

.

3.6 COMMUNICATION REQUIREMENTS contains the requirements that the CPDLC application imposes on the underlying communication system.

3.7 CPDLC USER REQUIREMENTS contains requirements imposed on the user of the CPDLC ASE service and message description tables.

3.8 SUBSETTING RULES defines the conformant subsets for the CPDLC-ASE.

Note.— throughout this document, reference to the CPDLC message refers to the actual CPDLC message, as generated by and delivered at the CPDLC users (operational content of the CPDLC message). Reference to CPDLC/IC data refers to CPDLC data, to which an integrity check (IC) (checksum) has been added before sending the CPDLC message (CPDLC/IC).

3.1.2 Functional Descriptions

3.1.2.1 The Controller-Pilot Message Exchange Function defines a method for a controller and pilot to exchange CPDLC messages via data link. This function provides CPDLC messages for the following :

a) general information exchange;

b) clearance

1) delivery,

2) request, and

3) response;

c) altitude/identity surveillance;

d) monitoring of current/planned position;

e) advisories

1) request, and

2) delivery;

f) system management functions; and

g) emergency situations.

3.1.2.2 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.

3.1.2.3 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).

3.1.2.4 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. The CPDLC Ground Forward function does not use the CPDLC Integrity Check.

3.2 GENERAL REQUIREMENTS

3.2.1 CPDLC ASE Version Number

3.2.1.1 The CPDLC-air-ASE and CPDLC-ground-ASE version numbers shall both be set to one.

3.2.2 Error Processing Requirements

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.

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 3.5, the CPDLC-user shall be notified.

3.3 THE ABSTRACT SERVICE

3.3.1 Service Description

3.3.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service shall exhibit external behaviour 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.

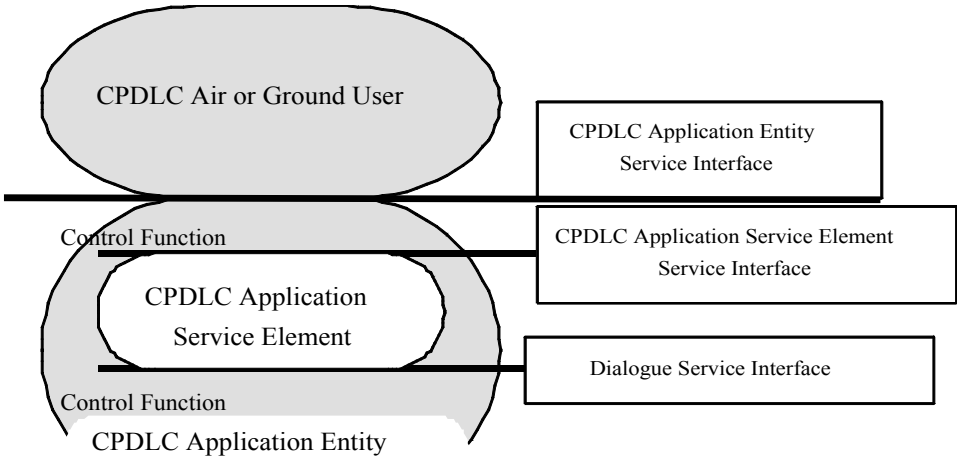


Figure 3.3-1. Function Model of the CPDLC Application

Note 3.— This chapter defines the static behaviour (i.e. the format) of the CPDLC abstract service. Its dynamic behaviour (i.e. how it is used) is described in 3.7.

Note 4.— Figure 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).

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, CPDLC Message creation, timer management, error and exception handling. The actual encoding and decoding of each CPDLC Message is handled by the CPDLC-users.

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.

3.3.2 The CPDLC-ASE Abstract Service

3.3.2.1 The CPDLC-ASE abstract service shall consist of a subset of the following services as allowed in 3.8:

a) CPDLC-start service as defined in 3.3.3,

b) DSC-start service as defined in 3.3.4,

c) CPDLC-message service as defined in 3.3.5,

d) CPDLC-end service as defined in 3.3.6,

e) DSC-end service as defined in 3.3.7,

f) CPDLC-forward service as defined in 3.3.8,

g) CPDLC-user-abort service as defined in 3.3.9, and

h) CPDLC-provider-abort service as defined in 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 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 3.4 for this field.

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.)

3.3.3.1 The CPDLC-start service shall contain the primitives and parameters as presented in Table 3.3-1.

Table 3.3-1. CPDLC-start Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Called Peer Identifier	M			
Calling Peer Identifier	M	M(=)		
CPDLC/IC Data	M	M(=)		
Response CPDLC/IC Data			M	M(=)
Result			M	M(=)
Class of Communication Service	U	M		
Security Required	U			

3.3.3.2 Called Peer Identifier

Note 1.— If the service is ground initiated, this parameter contains the addressed ICAO 24 bit aircraft address.

Note 2.— If the service is air initiated, this parameter contains the addressed ground system's facility designation.

3.3.3.2.1 If the service is ground initiated, the Called Peer Identifier parameter value shall conform to the abstract syntax of the ICAO 24-bit aircraft address.

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.

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 ICAO 24 bit aircraft address.

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.

3.3.3.3.2 If the service is air initiated, the *Calling Peer Identifier* parameter value shall conform to the abstract syntax of the ICAO 24-bit aircraft address.

3.3.3.4 CPDLC/IC Data

Note.— The CPDLC-user uses this parameter to send CPDLC/IC Data to its peer user.

3.3.3.4.1 If the CPDLC-start request primitive is invoked by the CPDLC-ground-user, the *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICUplinkMessage.

3.3.3.4.2 If the CPDLC-start request primitive is invoked by the CPDLC-air-user, the *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICDownlinkMessage.

Note.— The CPDLC/IC Data parameter usually contains an embedded operational CPDLC Message, but this is not present in some cases. It always contains an integrity check (IC) field.

3.3.3.5 Response CPDLC/IC Data

Note.— This parameter is used to either provide a reason for rejecting a CPDLC dialogue or to provide a means of verifying that a CPDLC dialogue establishment request has been received and accepted by the intended peer. In the latter case, the response includes either a LACK or no message element. This is to ensure a rapid response to a CPDLC Start. However, the CPDLC-ASE is not required to police this requirement. The air or ground CPDLC-user is instead required to ensure that the Response parameter includes either an allowed CPDLC Message or no CPDLC Message.

3.3.3.5.1 If the CPDLC-start response primitive is invoked by the CPDLC-ground-user, the *Response CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICUplinkMessage.

3.3.3.5.2 If the CPDLC-start response primitive is invoked by the CPDLC-air-user, the *Response CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICDownlinkMessage.

Note.— A Response may, and often does, contain no CPDLC Message. It always includes an integrity check (IC) field.

3.3.3.6 Result

Note.— This parameter is used to indicate whether or not a requested CPDLC dialogue is accepted.

3.3.3.6.1 This parameter shall have one of two abstract values: “accepted” or “rejected”.

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 the peer CPDLC user is that provided by the CPDLC user if specified by the user, else it indicates that no routing preference was requested by the CPDLC dialogue initiator.

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”.

3.3.3.7.2 When this parameter is provided by the CPDLC-user, the same value shall be indicated to the peer CPDLC user, else the abstract value “ATSC - No Traffic Type Policy Preference” is indicated.

3.3.3.8 Security Required

Note 1.— The Security Required *parameter contains the value of the required level of security, if specified by the CPDLC user.*

Note 2.— If the received Security Required *parameter is not as expected per the local security policy, the receiving CPDLC-ASE will abort.*

3.3.3.8.1 Where specified by the CPDLC-user, the *Security Required* parameter shall have one of the following abstract values: “no security” or “secured exchange”.

Note.— Where not specified by the CPDLC-user, this indicates that there is no security required.

3.3.4 DSC-start Service

Note 1.— The DSC-start service is used by the CPDLC-air-user 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.)

3.3.4.1 The DSC-start service shall contain the primitives and parameters as presented in Table 3.3-2.

Table 3.3-2. DSC-start Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Facility Designation	M			
Aircraft Address	M	M(=)		
CPDLC/IC Data	M	M(=)		
Response CPDLC/IC Data			M	M(=)
Result			M	M(=)
Class of Communication Service	U	M		
Security Required	U			

3.3.4.2 Facility Designation

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

3.3.4.2.1 The *Facility Desi*gnation parameter value shall conform to the abstract syntax four to eight-character facility designation.

3.3.4.3 Aircraft Address

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

Note.— This parameter contains the ICAO 24 bit aircraft address.

3.3.4.4 CPDLC/IC Data

Note.— The CPDLC-air-user uses this parameter to send CPDLC/IC Data to a CPDLC-ground-user.

3.3.4.4.1 The *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICDownlinkMessage.

Note.— The CPDLC/IC Data parameter usually contains an embedded operational CPDLC Message, but this is not present in some cases. It always contains an integrity check (IC) field.

3.3.4.5 Response CPDLC/IC Data

Note.— This parameter is used to either provide a reason for rejecting a DSC dialogue or to provide a means of verifying that a DSC dialogue establishment has been received and accepted by the intended peer. In the latter case, the response includes either a LACK or no message element. This is to ensure a rapid response

to a DSC Start. However, the CPDLC-ground-ASE is not required to police this requirement. The CPDLC-ground-user is instead required to ensure that the Response parameter includes either a LACK or no CPDLC Message.

3.3.4.5.1 The *Response CPDLC/IC Data* parameter shall conform to the ASN.1 abstract syntax ICUplinkMessage.

Note.— A Response may, and often does contain no CPDLC Message. It always includes an integrity check (IC) field.

3.3.4.6 Result

Note.— This parameter is used to indicate whether or not a requested DSC dialogue is accepted.

3.3.4.6.1 The *Result* parameter value shall have one of two abstract values: "accepted" or "rejected".

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 DSC dialogue.

Note 3.— If provided by the user, the parameter indicated to the peer user is that provided by the user, else it indicates that no routing preference was requested by the CPDLC dialogue initiator.

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".

3.3.4.7.2 When this parameter is provided by the CPDLC-user, the same value shall be indicated to the peer CPDLC-user, else the abstract value "ATSC - No Traffic Type Policy Preference" is indicated.

3.3.4.8 Security Required

Note 1.— The Security Required *parameter contains the value of the required level of security, if specified by the CPDLC-air-user.*

Note 2.— If the received Security Required *parameter is not as expected per the local security policy, the receiving CPDLC-ground-ASE will abort.*

3.3.4.8.1 Where specified by the CPDLC-air-user, the *Security Required* parameter shall have one of the following abstract values: "no security" or "secured exchange".

Note.— Where not specified by the CPDLC-air-user, this indicates that there is no security required.

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.

3.3.5.1 The CPDLC-message service shall contain the primitives and parameters as presented in Table .3.3-3.

Table 3.3-3. CPDLC-message Service Parameters

Parameter Name	Req	Ind
CPDLC/IC Data	M	M(=)

3.3.5.2 CPDLC/IC Data

Note.— This parameter contains a CPDLC/IC Data value.

3.3.5.2.1 If the CPDLC-message service is invoked by the CPDLC-ground-user, the *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICUplinkMessage.

3.3.5.2.2 If the CPDLC-message service is invoked by the CPDLC-air-user, the *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICDownlinkMessage.

Note.— This CPDLC/IC Data parameter provided by the CPDLC-user always contains both an embedded operational CPDLC Message and an integrity check field.

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.

3.3.6.1 The CPDLC-end service shall contain the primitives and parameters as presented in Table 3.3-4.

Table 3.3-4. CPDLC-end Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
CPDLC/IC Data	M	M(=)	M	M(=)
Result			M	M(=)

3.3.6.2 CPDLC/IC Data

Note.— This parameter contains a CPDLC/IC Data value.

3.3.6.2.1 The *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICUplinkMessage, if provided by the CPDLC-ground-user.

3.3.6.2.2 The *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICDownlinkMessage, if provided by the CPDLC-air-user.

Note.— The CPDLC/IC Data parameter value does not necessarily contain an embedded operational CPDLC Message. It always include an integrity check (IC) field.

3.3.6.3 Result

Note.— This parameter is used to indicate whether or not a request to terminate a CPDLC dialogue is accepted.

3.3.6.3.1 The *Result* parameter shall have one of two abstract values: “accepted” or “rejected”.

3.3.7 DSC-end Service

Note.— The DSC-end service is used by the CPDLC-air-user to end a DSC dialogue with a CPDLC-ground-user. It is a confirmed service.

3.3.7.1 The DSC-end service shall contain the primitives and parameters as presented in Table 3.3-5.

Table 3.3-5. DSC-end Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
CPDLC/IC Data	M	M(=)	M	M(=)
Result			M	M(=)

3.3.7.2 CPDLC/IC Data

Note.— This parameter contains a CPDLC/IC Data value.

3.3.7.2.1 The *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICUplinkMessage, if provided by the CPDLC-ground-user.

3.3.7.2.2 The *CPDLC/IC Data* parameter value shall conform to the ASN.1 abstract syntax ICDownlinkMessage if provided by the CPDLC-air-user.

Note.— The CPDLC/IC Data parameter value does not necessarily contain an embedded operational CPDLC Message. It always includes an integrity check (IC) field.

3.3.7.3 Result

Note.— This parameter is used to indicate whether or not a request to terminate a DSC dialogue is accepted.

3.3.7.3.1 The *Result* parameter shall have one of two abstract values: "accepted" or "rejected".

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.

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 3.3-6.

Table 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		
Security Required	U		
Result			M

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 3.3-7.

Table 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	
Security Required	U	
Result		M

3.3.8.3 Called Facility Designation

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

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

3.3.8.4 Calling Facility Designation

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

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

3.3.8.5 ASE Version Number

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

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.

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.

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.

3.3.8.6.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCForwardMessage.

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.

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”.

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.

3.3.8.8.1 The *Result* parameter value shall conform to the ASN.1 abstract syntax ATCForwardResponse.

3.3.8.9 Security Required

Note 1.— The Security Required *parameter contains the value of the required level of security, if specified by the CPDLC-ground-user.*

Note 2.— If the received Security Required *parameter is not as expected per the local security policy, the receiving CPDLC-ground-ASE will abort.*

3.3.8.9.1 Where specified by the CPDLC-ground-user, the *Security Required* parameter shall have one of the following abstract values: “no security” or “secured exchange”.

Note.— Where not specified by the CPDLC-ground-user, this indicates that there is no security required.

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 CPDLC-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.

3.3.9.1 The CPDLC-user-abort service shall contain the primitives and parameters as presented in Table 3.3-8.

Table 3.3-8. CPDLC-user-abort Service Parameters

Parameter Name	Req	Ind
Reason	U	M

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 CPDLC-user, the parameter indicated to the peer CPDLC-user is that provided by the CPDLC-user, else it is what the ASE supplies.

3.3.9.2.1 The *Reason* parameter value shall conform to the ASN.1 abstract syntax CPDLCUserAbortReason.

3.3.9.2.2 When this parameter is provided by the CPDLC-user, the same value shall be indicated to the peer CPDLC-user.

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.

3.3.10.1 The CPDLC-provider-abort service shall contain the primitives and parameters as presented in Table 3.3-9.

Table 3.3-9. CPDLC-provider-abort Service Parameters

Parameter Name	**Ind**
Reason	M

3.3.10.2 Reason

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

3.3.10.2.1 The *Reason* parameter shall conform to the ASN.1 abstract syntax CPDLCProviderAbortReason.

— — — — — — — —

Chapter 3

(Section 3.4)

(See mapping table for conversion of current paragraph numbers of Doc 9705 – 3rd edition into paragraph numbers of Doc 9880)

3.4 FORMAL DEFINITIONS OF MESSAGES

3.4.1 Encoding/Decoding Rules

3.4.1.1 A CPDLC-air-ASE shall be capable of encoding AircraftPDUs APDUs and decoding GroundPDUs APDUs as defined in the ASN.1 module CPDLCAPDUsVersion1 specified in 3.4.2.

Note.— The ICUplinkMessage and ICDownlinkMessage ASN.1 types both contain an integrity check (IC) field and the EncodedCPDLCMessage type, which resolves to a BIT STRING type. Even though this BIT STRING is specified as containing a PER Encoded ATCUplinkMessage or a PER Encoded ATCDownlinkMessage, there is no requirement on the CPDLC-air-ASE to verify this. The CPDLC-air-user is responsible for encoding a valid ATCDownlinkMessage and for verifying the correctness of a received ATCUplinkMessage.

3.4.1.2 The CPDLC-air-user shall be capable of encoding ATCDownlinkMessage and decoding ATCUplinkMessage types, as defined in the ASN.1 module CPDLCMessageSetVersion1 specified in 3.4.3.

3.4.1.3 A CPDLC-ground-ASE shall be capable of encoding GroundPDUs APDUs and decoding AircraftPDUs APDUs as defined in the ASN.1 module CPDLCAPDUsVersion1 specified in 3.4.2.

Note 1.— The ICUplinkMessage and ICDownlinkMessage ASN.1 types both contain an integrity check (IC) field and the CPDLCMessage type, which resolves to a BIT STRING type. Even though this BIT STRING is specified as containing a PER Encoded ATCUplinkMessage or a PER Encoded ATCDownlinkMessage, there is no requirement on the CPDLC-ground-ASE to verify this. The CPDLC-ground-user is responsible for encoding a valid ATCUplinkMessage and for verifying the correctness of a received ATCDownlinkMessage.

Note 2.— The ForwardMessage ASN.1 type contains the uplink or downlink CPDLC Message Element data type, which both resolve to a BIT STRING type. Even though this BIT STRING is specified as containing a PER Encoded ATCUplinkMessageData or a PER Encoded ATCDownlinkMessageData, there is no requirement on the CPDLC-ground-ASE to verify this. The CPDLC-ground-user is responsible for encoding and verifying the correctness of valid ATCDownlinkMessageData and ATCUplinkMessageData.

3.4.1.4 The CPDLC-ground-user shall be capable of encoding ATCUplinkMessage and decoding ATCDownlinkMessage types, as defined in the ASN.1 module CPDLCMessageSetVersion1 specified in 3.4.3.

3.4.1.5 The CPDLC-ground-user shall be capable of encoding and decoding ATCUplinkMessageData and ATCDownlinkMessageData types, as defined in the ASN.1 module CPDLCMessageSetVersion1 specified in 3.4.3.

3.4.2 CPDLC Message ASN.1 Abstract Syntax

3.4.2.1 The abstract syntax of the CPDLC protocol data units shall comply with the description contained in the ASN.1 module CPDLCAPDUsVersion1 conforming to ISO/IEC 8824, as defined in this section.

```
CPDLCAPDUsVersion1 DEFINITIONS::=

BEGIN

IMPORTS
        DateTimeGroup,
        AircraftFlightIdentification,
        AircraftAddress
        FROM CPDLCMessageSetVersion1;

-- ---------------------------------------------------------------------------

-- Ground Generated Messages - Top level

-- ---------------------------------------------------------------------------

GroundPDUs ::= CHOICE

        {

        abortUser           [0]     CPDLCUserAbortReason,

        abortProvider       [1]     CPDLCProviderAbortReason,

        startup             [2]     ICUplinkMessage,

        send                [3]     ICUplinkMessage,

        forward             [4]     ATCForwardMessage,

        forwardresponse     [5]     ATCForwardResponse,

        ...

        }
```

```
ICUplinkMessage<blank>::= SEQUENCE
        {
        algorithmIdentifier     [0] AlgorithmIdentifier OPTIONAL,
        embeddedMessage         [1] EncodedCPDLCMessage OPTIONAL,
                                        -- PER encoded ATCUplinkMessage
                                        -- (see Module CPDLCMessageSetVersion1)
        integrityCheck          [2] BIT STRING,
        ...
}

ATCForwardMessage ::= SEQUENCE
        {
        forwardHeader           ForwardHeader,
        forwardMessage          ForwardMessage
        }

ForwardHeader ::= SEQUENCE
        {
        dateTime                DateTimeGroup,
        aircraftID              AircraftFlightIdentification,
        aircraftAddress         AircraftAddress
        }
```

```
ForwardMessage ::= CHOICE
        {
        upElementIDs        [0]     BIT STRING,
                                        --PER encoded ATCUplinkMessageData,
                                        -- (see Module CPDLCMessageSetVersion1)
        downElementIDs      [1]     BIT STRING
                                        --PER encoded ATCDownlinkMessageData,
                                        -- (see Module CPDLCMessageSetVersion1)     }

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]     ICDownlinkMessage,
        ...
        }

StartDownMessage ::= SEQUENCE
        {
        mode                    Mode DEFAULT cpdlc,
        startDownlinkMessage    ICDownlinkMessage
        }

Mode ::= ENUMERATED
        {
        cpdlc           (0),
        dsc             (1)
        }


ICDownlinkMessage<blank>::= SEQUENCE
        {
        algorithmIdentifier             [0] AlgorithmIdentifier OPTIONAL,
        embeddedMessage                 [1] EncodedCPDLCMessage OPTIONAL,
                                                --PER encoded ATCDownlinkMessage,
                                                -- (see Module CPDLCMessageSetVersion1)
        integrityCheck                  [2] BIT STRING,
        ...
        }
```

```
-- ---------------------------------------------------------------------------
-- Uplink and Downlink messages - Common Elements

-- ---------------------------------------------------------------------------

AlgorithmIdentifier ::= RELATIVE-OID --root is {icao-arc atn-algorithms(9)}

                                        -- see Doc 9705 Sub-Volume IX for OID root assignment


EncodedCPDLCMessage ::= BIT STRING


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),

        time-out-of-synchronisation                 (7),

        unknown-integrity-check                     (8),

        validation-failure                          (9),

        unable-to-decode-message                    (10),

        invalid-pdu                                 (11),

        invalid-CPDLC-message                       (12),

        ...

        }
```

```
CPDLCProviderAbortReason ::= ENUMERATED
        {
        timer-expired                          (0),
        undefined-error                        (1),\
        invalid-PDU                            (2),
        protocol-error                         (3),
        communication-service-error            (4),
        communication-service-failure          (5),
        invalid-QOS-parameter                  (6),
        expected-PDU-missing                   (7),
        ...
        }

END
```

3.4.3 CPDLC Message ASN.1 Abstract Syntax

3.4.3.1 The abstract syntax of the CPDLC messages shall comply with the description contained in the ASN.1 module CPDLCMessageSetVersion1 conforming to ISO/IEC 8824, as defined in this section.

Note.— The object identifier "cpdlc-message-AS-v1" as defined below identifies the CPDLC Message Set Version 1:

cpdlc-message-AS-v1 ::= OBJECT IDENTIFIER {iso(1) identified organisation (3) icao(27) user message abstract syntax (10) cpdlc (1) version1 (1)

```
CPDLCMessageSetVersion1 DEFINITIONS::=

BEGIN

ATCUplinkMessage ::= SEQUENCE
        {
        header                  ATCMessageHeader,
        messageData             ATCUplinkMessageData
        }

ATCUplinkMessageData ::= SEQUENCE
        {
        elementIds              SEQUENCE SIZE (1..5) OF ATCUplinkMsgElementId,
        constrainedDataSEQUENCE
                {
                routeClearanceData      SEQUENCE SIZE (1..2) OF RouteClearance OPTIONAL,
                ...
                }
                OPTIONAL
        }
ATCDownlinkMessage ::= SEQUENCE
        {
        header                  ATCMessageHeader,
        messageData             ATCDownlinkMessageData
        }
```

```
ATCDownlinkMessageData ::= SEQUENCE
        {
        elementIds              SEQUENCE SIZE (1..5) OF ATCDownlinkMsgElementId,
        constrainedDataSEQUENCE
                {
                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)
```

```
LogicalAck ::= ENUMERATED
        {
        required                    (0),
        notRequired                 (1)
        }
-- ---------------------------------------------------------------------------
-- Uplink message element
-- ---------------------------------------------------------------------------

ATCUplinkMsgElementId ::= CHOICE
      {
--    UNABLE                                    Urg(N)/Alr(M)/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]	Urg(L)/Alr(L)/Resp(R)
	uM9Time	[9] Time,
--	EXPECT DESCENT AT [position]	Urg(L)/Alr(L)/Resp(R)
	uM10Position	[10] Position,
--	EXPECT CRUISE CLIMB AT [time]	Urg(L)/Alr(L)/Resp(R)
	uM11Time	[11] Time,

--	EXPECT CRUISE CLIMB AT [position]	Urg(L)/Alr(L)/Resp(R)
	uM12Position	[12] Position,
--	AT [time] EXPECT CLIMB TO [level]	Urg(L)/Alr(L)/Resp(R)
	uM13TimeLevel	[13] TimeLevel,
--	AT [position] EXPECT CLIMB TO [level]	Urg(L)/Alr(L)/Resp(R)
	uM14PositionLevel	[14] PositionLevel,
--	AT [time] EXPECT DESCENT TO [level]	Urg(L)/Alr(L)/Resp(R)
	uM15TimeLevel	[15] TimeLevel,
--	AT [position] EXPECT DESCENT TO [level]	Urg(L)/Alr(L)/Resp(R)
	uM16PositionLevel	[16] PositionLevel,
--	AT [time] EXPECT CRUISE CLIMB TO [level]	Urg(L)/Alr(L)/Resp(R)
	uM17TimeLevel	[17] TimeLevel,
--	AT [position] EXPECT CRUISE CLIMB TO [level]	Urg(L)/Alr(L)/Resp(R)
	uM18PositionLevel	[18] PositionLevel,
--	MAINTAIN [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM19Level	[19] Level,

--	CLIMB TO [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM20Level	[20] Level,
--	AT [time] CLIMB TO [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM21TimeLevel	[21] TimeLevel,
--	AT [position] CLIMB TO [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM22PositionLevel	[22] PositionLevel,
--	DESCEND TO [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM23Level	[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(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]             Urg(N)/Alr(M)/Resp(W/U)
        uM95DirectionDegrees                                [95] DirectionDegrees,

--      CONTINUE PRESENT HEADING                            Urg(N)/Alr(M)/Resp(W/U)
        uM96NULL                                            [96] NULL,

--      AT [position] FLY HEADING [degrees]                 Urg(N)/Alr(M)/Resp(W/U)
        uM97PositionDegrees                                 [97] PositionDegrees,

--      IMMEDIATELY TURN [direction] HEADING [degrees]
--                                                          Urg(D)/Alr(H)/Resp(W/U)
        uM98DirectionDegrees                                [98] DirectionDegrees,

--      EXPECT [procedureName]                              Urg(L)/Alr(L)/Resp(R)
        uM99ProcedureName                                   [99] ProcedureName,

--      AT [time] EXPECT [speed]                            Urg(L)/Alr(L)/Resp(R)
        uM100TimeSpeed                                      [100] TimeSpeed,

--      AT [position] EXPECT [speed]                        Urg(L)/Alr(L)/Resp(R)
        uM101PositionSpeed                                  [101] PositionSpeed,
```

--	AT [level] EXPECT [speed]	Urg(L)/Alr(L)/Resp(R)
	uM102LevelSpeed	[102] LevelSpeed,
--	AT [time] EXPECT [speed] TO [speed]	Urg(L)/Alr(L)/Resp(R)
	uM103TimeSpeedSpeed	[103] TimeSpeedSpeed,
--	AT [position] EXPECT [speed] TO [speed]	Urg(L)/Alr(L)/Resp(R)
	uM104PositionSpeedSpeed	[104] PositionSpeedSpeed,
--	AT [level] EXPECT [speed] TO [speed]	Urg(L)/Alr(L)/Resp(R)
	uM105LevelSpeedSpeed	[105] LevelSpeedSpeed,
--	MAINTAIN [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM106Speed	[106] Speed,
--	MAINTAIN PRESENT SPEED	Urg(N)/Alr(M)/Resp(W/U)
	uM107NULL	[107] NULL,
--	MAINTAIN [speed] OR GREATER	Urg(N)/Alr(M)/Resp(W/U)
	uM108Speed	[108] Speed,
--	MAINTAIN [speed] OR LESS	Urg(N)/Alr(M)/Resp(W/U)
	uM109Speed	[109] Speed,

--	MAINTAIN [speed] TO [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM110SpeedSpeed	[110] SpeedSpeed,
--	INCREASE SPEED TO [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM111Speed	[111] Speed,
--	INCREASE SPEED TO [speed] OR GREATER	Urg(N)/Alr(M)/Resp(W/U)
	uM112Speed	[112] Speed,
--	REDUCE SPEED TO [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM113Speed	[113] Speed,
--	REDUCE SPEED TO [speed] OR LESS	Urg(N)/Alr(M)/Resp(W/U)
	uM114Speed	[114] Speed,
--	DO NOT EXCEED [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM115Speed	[115] Speed,
--	RESUME NORMAL SPEED	Urg(N)/Alr(M)/Resp(W/U)
	uM116NULL	[116] NULL,

--	CONTACT [unitname] [frequency]	Urg(N)/Alr(M)/Resp(W/U)
	uM117UnitNameFrequency	[117] UnitNameFrequency,
--	AT [position] CONTACT [unitname] [frequency]	Urg(N)/Alr(M)/Resp(W/U)
	uM118PositionUnitNameFrequency	[118] PositionUnitNameFrequency,
--	AT [time] CONTACT [unitname] [frequency]	Urg(N)/Alr(M)/Resp(W/U)
	uM119TimeUnitNameFrequency	[119] TimeUnitNameFrequency,
--	MONITOR [unitname] [frequency]	Urg(N)/Alr(M)/Resp(W/U)
	uM120UnitNameFrequency	[120] UnitNameFrequency,
--	AT [position] MONITOR [unitname] [frequency]	Urg(N)/Alr(M)/Resp(W/U)
	uM121PositionUnitNameFrequency	[121] PositionUnitNameFrequency,
--	AT [time] MONITOR [unitname] [frequency]	Urg(N)/Alr(M)/Resp(W/U)
	uM122TimeUnitNameFrequency	[122] TimeUnitNameFrequency,
--	SQUAWK [code]	Urg(N)/Alr(M)/Resp(W/U)
	uM123Code	[123] Code,
--	STOP SQUAWK	Urg(N)/Alr(M)/Resp(W/U)
	uM124NULL	[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(L)/Resp(W/U)
        uM127NULL                               [127] NULL,

--      REPORT LEAVING [level]                  Urg(N)/Alr(L)/Resp(W/U)
        uM128Level                              [128] Level,

--      REPORT MAINTAINING [level]              Urg(N)/Alr(L)/Resp(W/U)
        uM129Level                              [129] Level,

--      REPORT PASSING [position]               Urg(N)/Alr(L)/Resp(W/U)
        uM130Position                           [130] Position,

--      REPORT REMAINING FUEL AND PERSONS ON BOARD
--                                              Urg(U)/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(L)/Resp(Y)
	uM135NULL	[135] NULL,
--	CONFIRM ASSIGNED SPEED	Urg(N)/Alr(L)/Resp(Y)
	uM136NULL	[136] NULL,
--	CONFIRM ASSIGNED ROUTE	Urg(N)/Alr(L)/Resp(Y)
	uM137NULL	[137] NULL,
--	CONFIRM TIME OVER REPORTED WAYPOINT	Urg(N)/Alr(L)/Resp(Y)
	uM138NULL	[138] NULL,
--	CONFIRM REPORTED WAYPOINT	Urg(N)/Alr(L)/Resp(Y)
	uM139NULL	[139] NULL,

```
--      CONFIRM NEXT WAYPOINT                   Urg(N)/Alr(L)/Resp(Y)
        uM140NULL                               [140] NULL,

--      CONFIRM NEXT WAYPOINT ETA               Urg(N)/Alr(L)/Resp(Y)
        uM141NULL                               [141] NULL,

--      CONFIRM ENSUING WAYPOINT                Urg(N)/Alr(L)/Resp(Y)
        uM142NULL                               [142] NULL,

--      CONFIRM REQUEST                         Urg(N)/Alr(L)/Resp(Y)
        uM143NULL                               [143] NULL,

--      CONFIRM SQUAWK                          Urg(N)/Alr(L)/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(L)/Resp(Y)
        uM148Level                                      [148] Level,

--      CAN YOU ACCEPT [level] AT [position]            Urg(N)/Alr(L)/Resp(A/N)
        uM149LevelPosition                              [149] LevelPosition,

--      CAN YOU ACCEPT [level] AT [time]                Urg(N)/Alr(L)/Resp(A/N)
        uM150LevelTime                                  [150] LevelTime,
--      WHEN CAN YOU ACCEPT [speed]                     Urg(N)/Alr(L)/Resp(Y)
        uM151Speed                                      [151] Speed,

--      WHEN CAN YOU ACCEPT [specifiedDistance] [direction] OFFSET
--                                                      Urg(N)/Alr(L)/Resp(Y)
        uM152DistanceSpecifiedDirection                 [152] DistanceSpecifiedDirection,

--      ALTIMETER [altimeter]                           Urg(N)/Alr(L)/Resp(R)
        uM153Altimeter                                  [153] Altimeter,

--      RADAR SERVICE TERMINATED                        Urg(N)/Alr(L)/Resp(R)
        uM154NULL                                       [154] NULL,

--      RADAR CONTACT [position]                        Urg(N)/Alr(M)/Resp(R)
        uM155Position                                   [155] Position,
```

--	RADAR CONTACT LOST	Urg(N)/Alr(M)/Resp(R)
	uM156NULL	[156] NULL,
--	CHECK STUCK MICROPHONE [frequency]	Urg(U)/Alr(M)/Resp(N)
	uM157Frequency	[157] Frequency,
--	ATIS [atiscode]	Urg(N)/Alr(L)/Resp(R)
	uM158AtisCode	[158] ATISCode,
--	ERROR [errorInformation]	Urg(U)/Alr(M)/Resp(N)
	uM159ErrorInformation	[159] ErrorInformation,
--	NEXT DATA AUTHORITY [facility]	Urg(L)/Alr(N)/Resp(N)
	uM160Facility	[160] Facility,
--	END SERVICE	Urg(L)/Alr(N)/Resp(N)
	uM161NULL	[161] NULL,
--	SERVICE UNAVAILABLE	Urg(L)/Alr(L)/Resp(N)
	uM162NULL	[162] NULL,
--	[facilitydesignation]	Urg(L)/Alr(N)/Resp(N)
	uM163FacilityDesignation	[163] FacilityDesignation,

--	WHEN READY	Urg(L)/Alr(N)/Resp(N)
	uM164NULL	[164] NULL,
--	THEN	Urg(L)/Alr(N)/Resp(N)
	uM165NULL	[165] NULL,
--	DUE TO [traffictype]TRAFFIC	Urg(L)/Alr(N)/Resp(N)
	uM166TrafficType	[166] TrafficType,
--	DUE TO AIRSPACE RESTRICTION	Urg(L)/Alr(N)/Resp(N)
	uM167NULL	[167] NULL,
--	DISREGARD	Urg(U)/Alr(M)/Resp(R)
	uM168NULL	[168] NULL,
--	[freetext]	Urg(N)/Alr(L)/Resp(R)
	uM169FreeText	[169] FreeText,
--	[freetext]	Urg(D)/Alr(H)/Resp(R)
	uM170FreeText	[170] FreeText,
--	CLIMB AT [verticalRate] MINIMUM	Urg(N)/Alr(M)/Resp(W/U)
	uM171VerticalRate	[171] VerticalRate,

--	CLIMB AT [verticalRate] MAXIMUM	Urg(N)/Alr(M)/Resp(W/U)
	uM172VerticalRate	[172] VerticalRate,
--	DESCEND AT [verticalRate] MINIMUM	Urg(N)/Alr(M)/Resp(W/U)
	uM173VerticalRate	[173] VerticalRate,
--	DESCEND AT [verticalRate] MAXIMUM	Urg(N)/Alr(M)/Resp(W/U)
	uM174VerticalRate	[174] VerticalRate,
--	REPORT REACHING [level]	Urg(N)/Alr(L)/Resp(W/U)
	uM175Level	[175] Level,
--	MAINTAIN OWN SEPARATION AND VMC	Urg(N)/Alr(M)/Resp(W/U)
	uM176NULL	[176] NULL,
--	AT PILOTS DISCRETION	Urg(L)/Alr(L)/Resp(N)
	uM177NULL	[177] NULL,
--	Reserved	Urg(L)/Alr(L)/Resp(Y)
	uM178NULL	[178] NULL,
--	SQUAWK IDENT	Urg(N)/Alr(M)/Resp(W/U)
	uM179NULL	[179] NULL,

--	REPORT REACHING BLOCK [level] TO [level]	Urg(N)/Alr(L)/Resp(W/U)
	uM180LevelLevel	[180] LevelLevel,
--	REPORT DISTANCE [tofrom] [position]	Urg(N)/Alr(M)/Resp(Y)
	uM181ToFromPosition	[181] ToFromPosition,
--	CONFIRM ATIS CODE	Urg(N)/Alr(L)/Resp(Y)
	uM182NULL	[182] NULL,
--	[freetext]	Urg(N)/Alr(M)/Resp(N)
	uM183FreeText	[183] FreeText,
--	AT [time] REPORT DISTANCE [tofrom] [position]	Urg(N)/Alr(L)/Resp(Y)
	uM184TimeToFromPosition	[184] TimeToFromPosition,
--	AFTER PASSING [position] CLIMB TO [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM185PositionLevel	[185] PositionLevel,
--	AFTER PASSING [position] DESCEND TO [level]	Urg(N)/Alr(M)/Resp(W/U)
	uM186PositionLevel	[186] PositionLevel,
--	[freetext]	Urg(L)/Alr(N)/Resp(N)
	uM187FreeText	[187] FreeText,

--	AFTER PASSING [position] MAINTAIN [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM188PositionSpeed	[188] PositionSpeed,
--	ADJUST SPEED TO [speed]	Urg(N)/Alr(M)/Resp(W/U)
	uM189Speed	[189] Speed,
--	FLY HEADING [degrees]	Urg(N)/Alr(M)/Resp(W/U)
	uM190Degrees	[190] Degrees,
--	ALL ATS TERMINATED	Urg(N)/Alr(M)/Resp(R)
	uM191NULL	[191] NULL,
--	REACH [level] BY [time]	Urg(N)/Alr(M)/Resp(W/U)
	uM192LevelTime	[192] LevelTime,
--	IDENTIFICATION LOST	Urg(N)/Alr(M)/Resp(R)
	uM193NULL	[193] NULL,
--	[freetext]	Urg(N)/Alr(L)/Resp(Y)
	uM194FreeText	[194] FreeText,
--	[freetext]	Urg(L)/Alr(L)/Resp(R)
	uM195FreeText	[195] FreeText,

```
--      [freetext]                          Urg(N)/Alr(M)/Resp(W/U)
        uM196FreeText                       [196] FreeText,
--      [freetext]                          Urg(U)/Alr(M)/Resp(W/U)
        uM197FreeText                       [197] FreeText,

--      [freetext]                          Urg(D)/Alr(H)/Resp(W/U)
        uM198FreeText                       [198] FreeText,

--      [freetext]                          Urg(N)/Alr(L)/Resp(N)
        uM199FreeText                       [199] FreeText,

--      REPORT REACHING                     Urg(N)/Alr(L)/Resp(W/U)
        uM200NULL                           [200] NULL,

--      Not Used                            Urg(L)/Alr(L)/Resp(N)
        uM201NULL                           [201] NULL,

--      Not Used                            Urg(L)/Alr(L)/Resp(N)
        uM202NULL                           [202] NULL,

--      [freetext]                          Urg(N)/Alr(M)/Resp(R)
        uM203FreeText                       [203] FreeText,
```

--	[freetext]	Urg(N)/Alr(M)/Resp(Y)
	uM204FreeText	[204] FreeText,
--	[freetext]	Urg(N)/Alr(M)/Resp(A/N)
	uM205FreeText	[205] FreeText,
--	[freetext]	Urg(L)/Alr(N)/Resp(Y)
	uM206FreeText	[206] FreeText,
--	[freetext]	Urg(L)/Alr(L)/Resp(Y)
	uM207FreeText	[207] FreeText,
--	[freetext]	Urg(L)/Alr(L)/Resp(N)
	uM208FreeText	[208] FreeText,
--	REACH [level] BY [position]	Urg(N)/Alr(M)/Resp(W/U)
	uM209LevelPosition	[209] LevelPosition,
--	IDENTIFIED [position]	Urg(N)/Alr(M)/Resp(R)
	uM210Position	[210] Position,
--	REQUEST FORWARDED	Urg(N)/Alr(L)/Resp(N)
	uM211NULL	[211] NULL,

--	[facilitydesignation] ATIS [atiscode] CURRENT	Urg(N)/Alr(L)/Resp(R)
	uM212FacilityDesignationATISCode	[212] FacilityDesignationATISCode,
--	[facilitydesignation] ALTIMETER [altimeter]	Urg(N)/Alr(L)/Resp(R)
	uM213FacilityDesignationAltimeter	[213] FacilityDesignationAltimeter,
--	RVR RUNWAY [runway] [rvr]	Urg(N)/Alr(M)/Resp(R)
	uM214RunwayRVR	[214] RunwayRVR,
--	TURN [direction][degrees]	Urg(N)/Alr(M)/Resp(W/U)
	uM215DirectionDegrees	[215] DirectionDegrees,
--	REQUEST FLIGHT PLAN	Urg(N)/Alr(M)/Resp(Y)
	uM216NULL	[216] NULL,
--	REPORT ARRIVAL	Urg(N)/Alr(M)/Resp(Y)
	uM217NULL	[217] NULL,
--	REQUEST ALREADY RECEIVED	Urg(L)/Alr(N)/Resp(N)
	uM218NULL	[218] NULL,
--	STOP CLIMB AT [level]	Urg(U)/Alr(M)/Resp(W/U)
	uM219Level	[219] Level,

--	STOP DESCENT AT [level]	Urg(U)/Alr(M)/Resp(W/U)
	uM220Level	[220] Level,
--	STOP TURN HEADING [degrees]	Urg(U)/Alr(M)/Resp(W/U)
	uM221Degrees	[221] Degrees,
--	NO SPEED RESTRICTION	Urg(L)/Alr(L)/Resp(R)
	uM222NULL	[222] NULL,
--	REDUCE TO MINIMUM APPROACH SPEED	Urg(N)/Alr(M)/Resp(W/U)
	uM223NULL	[223] NULL,
--	NO DELAY EXPECTED	Urg(N)/Alr(L)/Resp(R)
	uM224NULL	[224] NULL,
--	DELAY NOT DETERMINED	Urg(N)/Alr(L)/Resp(R)
	uM225NULL	[225] NULL,
--	EXPECTED APPROACH TIME [time]	Urg(N)/Alr(L)/Resp(R)
	uM226Time	[226] Time,
--	LOGICAL ACKNOWLEDGEMENT	Urg(N)/Alr(M)/Resp(N)
	uM227NULL	[227] NULL,

```
--      REPORT ETA [position]                       Urg(L)/Alr(L)/Resp(Y)
        uM228Position                               [228] Position,

--      REPORT ALTERNATE AERODROME                  Urg(L)/Alr(L)/Resp(Y)
        uM229NULL                                   [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 ACKNOWLEDGEMENT 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,

--      REQUEST AGAIN WITH NEXT UNIT                    Urg(N)/Alr(L)/Resp(N)
         uM237NULL                                      [237] 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 CLEARANCE [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(M)/Resp(Y)
        dM26PositionRouteClearance                      [26] PositionRouteClearanceIndex,

--      REQUEST WEATHER DEVIATION UP TO [specifiedDistance] [direction] OF ROUTE
--                                                      Urg(N)/Alr(M)/Resp(Y)
        dM27DistanceSpecifiedDirection                  [27] DistanceSpecifiedDirection,

--      LEAVING [level]                                 Urg(N)/Alr(L)/Resp(N)
        dM28Level                                       [28] Level,

--      CLIMBING TO [level]                             Urg(N)/Alr(L)/Resp(N)
        dM29Level                                       [29]Level,
```

--	DESCENDING TO [level]	Urg(N)/Alr(L)/Resp(N)
	dM30Level	[30] Level,
--	PASSING [position]	Urg(N)/Alr(L)/Resp(N)
	dM31Position	[31] Position,
--	PRESENT LEVEL [level]	Urg(N)/Alr(L)/Resp(N)
	dM32Level	[32] Level,
--	PRESENT POSITION [position]	Urg(N)/Alr(L)/Resp(N)
	dM33Position	[33] Position,
--	PRESENT SPEED [speed]	Urg(N)/Alr(L)/Resp(N)
	dM34Speed	[34] Speed,
--	PRESENT HEADING [degrees]	Urg(N)/Alr(L)/Resp(N)
	dM35Degrees	[35] Degrees,
--	PRESENT GROUND TRACK [degrees]	Urg(N)/Alr(L)/Resp(N)
	dM36Degrees	[36] Degrees,
--	MAINTAINING [level]	Urg(N)/Alr(L)/Resp(N)
	dM37Level	[37] Level,

--	ASSIGNED LEVEL [level]	Urg(N)/Alr(M)/Resp(N)
	dM38Level	[38] Level,
--	ASSIGNED SPEED [speed]	Urg(N)/Alr(M)/Resp(N)
	dM39Speed	[39] Speed,
--	ASSIGNED ROUTE [routeClearance]	Urg(N)/Alr(M)/Resp(N)
	dM40RouteClearance	[40] RouteClearanceIndex,
--	BACK ON ROUTE	Urg(N)/Alr(M)/Resp(N)
	dM41NULL	[41] NULL,
--	NEXT WAYPOINT [position]	Urg(N)/Alr(L)/Resp(N)
	dM42Position	[42] Position,
--	NEXT WAYPOINT ETA [time]	Urg(N)/Alr(L)/Resp(N)
	dM43Time	[43] Time,
--	ENSUING WAYPOINT [position]	Urg(N)/Alr(L)/Resp(N)
	dM44Position	[44] Position,
--	REPORTED WAYPOINT [position]	Urg(N)/Alr(L)/Resp(N)
	dM45Position	[45] Position,

--	REPORTED WAYPOINT [time]	Urg(N)/Alr(L)/Resp(N)
	dM46Time	[46] Time,
--	SQUAWKING [code]	Urg(N)/Alr(L)/Resp(N)
	dM47Code	[47] Code,
--	POSITION REPORT [positionreport]	Urg(N)/Alr(M)/Resp(N)
	dM48PositionReport	[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(L)/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(L)/Resp(N)
        dM76LevelLevel                                      [76] LevelLevel,

--      ASSIGNED BLOCK [level] TO [level]                   Urg(N)/Alr(M)/Resp(N)
        dM77LevelLevel                                      [77] LevelLevel,

--      AT [time] [distance] [tofrom] [position]            Urg(N)/Alr(L)/Resp(N)
        dM78TimeDistanceToFromPosition                      [78] TimeDistanceToFromPosition,

--      ATIS [atiscode]                                     Urg(N)/Alr(L)/Resp(N)
        dM79AtisCode                                        [79] ATISCode,

--      DEVIATING UP TO [specifiedDistance] [direction] OF ROUTE
--                                                          Urg(U)/Alr(H)/Resp(Y)
        dM80DistanceSpecifiedDirection                      [80] DistanceSpecifiedDirection,

--      WE CAN ACCEPT [level] AT [time]                     Urg(L)/Alr(L)/Resp(N)
        dM81LevelTime                                       [81] LevelTime,

--      WE CANNOT ACCEPT [level]                            Urg(L)/Alr(L)/Resp(N)
        dM82Level                                           [82] Level,
```

```
--      WE CAN ACCEPT [speed] AT [time]                 Urg(L)/Alr(L)/Resp(N)
        dM83SpeedTime                                   [83] SpeedTime,

--      WE CANNOT ACCEPT [speed]                        Urg(L)/Alr(L)/Resp(N)
        dM84Speed                                       [84] Speed,

--      WE CAN ACCEPT [specifiedDistance] [direction] AT [time]
--                                                      Urg(L)/Alr(L)/Resp(N)
        dM85DistanceSpecifiedDirectionTime              [85] DistanceSpecifiedDirectionTime,

--      WE CANNOT ACCEPT [specifiedDistance] [direction]  Urg(L)/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(U)/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]	Urg(L)/Alr(L)/Resp(Y)
	dM92FreeText	[92] FreeText,
--	[freetext]	Urg(U)/Alr(H)/Resp(N)
	dM93FreeText	[93] FreeText,
--	[freetext]	Urg(D)/Alr(H)/Resp(N)
	dM94FreeText	[94] FreeText,
--	[freetext]	Urg(U)/Alr(M)/Resp(N)
	dM95FreeText	[95] FreeText,
--	[freetext]	Urg(U)/Alr(L)/Resp(N)
	dM96FreeText	[96] FreeText,
--	[freetext]	Urg(L)/Alr(L)/Resp(N)
	dM97FreeText	[97] FreeText,
--	[freetext]	Urg(N)/Alr(N)/Resp(N)
	dM98FreeText	[98] FreeText,

```
--      CURRENT DATA AUTHORITY                  Urg(L)/Alr(L)/Resp(N)
        dM99NULL                                [99] NULL,

--      LOGICAL ACKNOWLEDGEMENT                 Urg(N)/Alr(M)/Resp(N)
        dM100NULL                               [100] NULL,

--      REQUEST END OF SERVICE                  Urg(L)/Alr(L)/Resp(Y)
        dM101NULL                               [101] NULL,

--      LANDING REPORT                          Urg(N)/Alr(N)/Resp(N)
        dM102NULL                               [102] NULL,

--      CANCELLING IFR                          Urg(N)/Alr(L)/Resp(Y)
        dM103NULL                               [103] NULL,

--      ETA[position][time]                     Urg(L)/Alr(L)/Resp(N)
        dM104PositionTime                       [104] PositionTime,

--      ALTERNATE AERODROME[airport]            Urg(L)/Alr(L)/Resp(N)
        dM105Airport                            [105] Airport,

--      PREFERRED LEVEL[level]                  Urg(L)/Alr(L)/Resp(N)
        dM106Level                              [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]SpeedTypeSpeedTypeSpeedTypeSpeed,
        ...
        }
```

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),
    logicalACKNOWLEDGEMENTNotAccepted                   (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
        }

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 (118.000..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
    {
    reserved        (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,
        latLonWholeMinutes                  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]     LongitudeDegrees,
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
    }

PositionRouteClearanceIndex ::= SEQUENCE
    {
    position                Position,
    routeClearanceIndex     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

```
{
airportDepartture          [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
```

Chapter 3

(Section 3.5)

(See mapping table for conversion of current paragraph numbers of Doc 9705 – 3rd edition into paragraph numbers of Doc 9880)

3.5 PROTOCOL DEFINITION

3.5.1 Sequence Rules

3.5.1.1 With the exception of abort primitives, only the sequence of primitives illustrated in Figures 3.5-1 to 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.

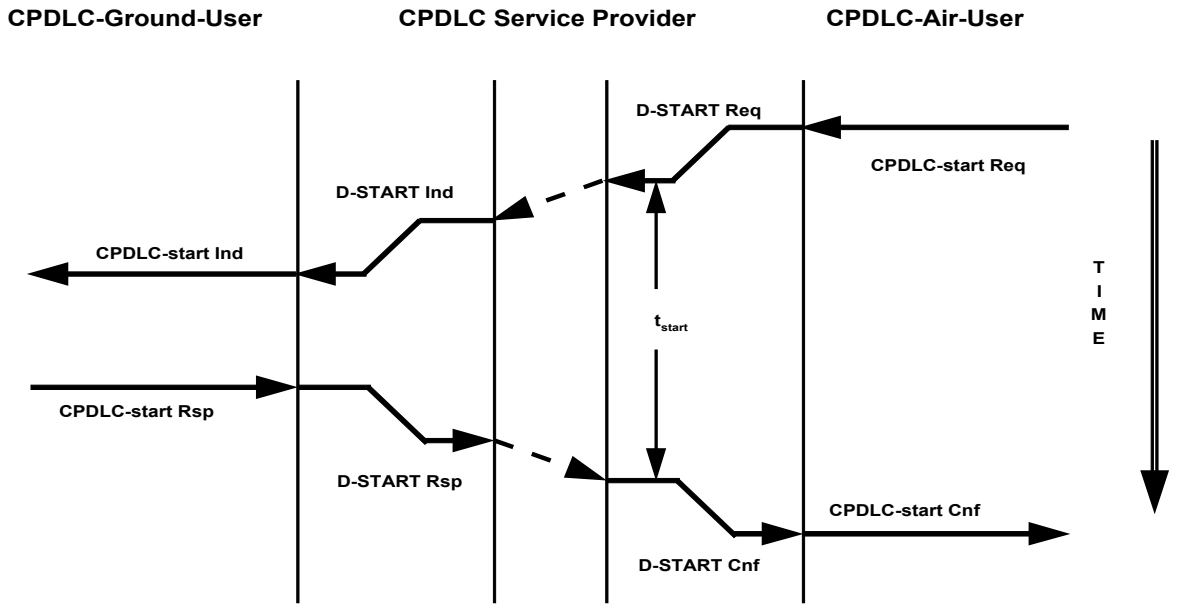


Figure 3.5-1. Sequence Diagram for CPDLC-start Service/Air Initiated

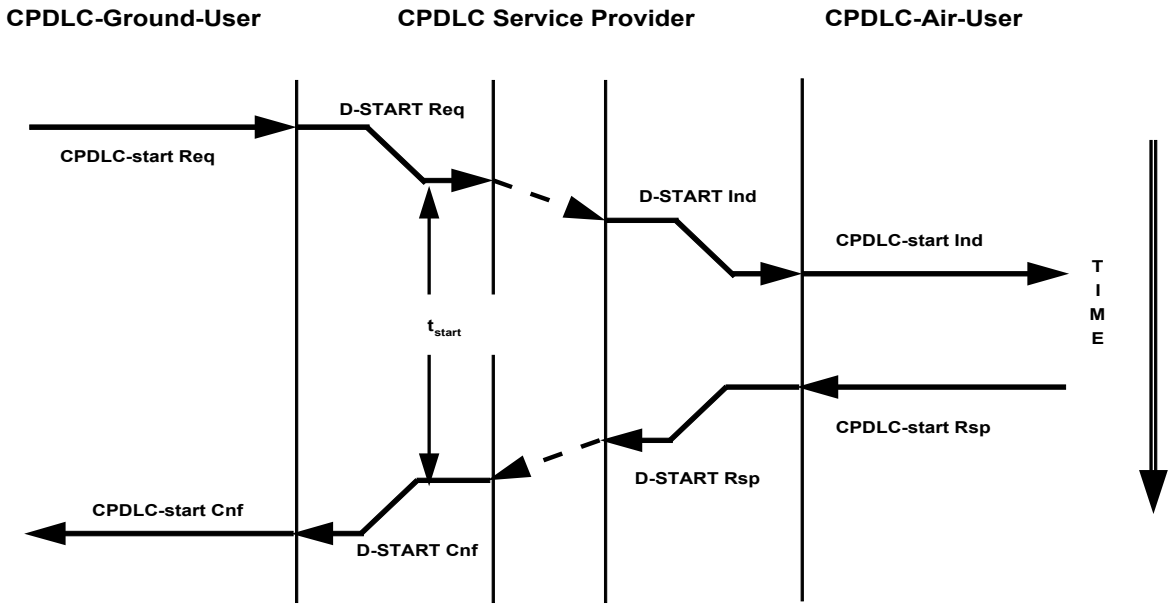


Figure 3.5-2. Sequence Diagram for CPDLC-start Service/Ground Initiated

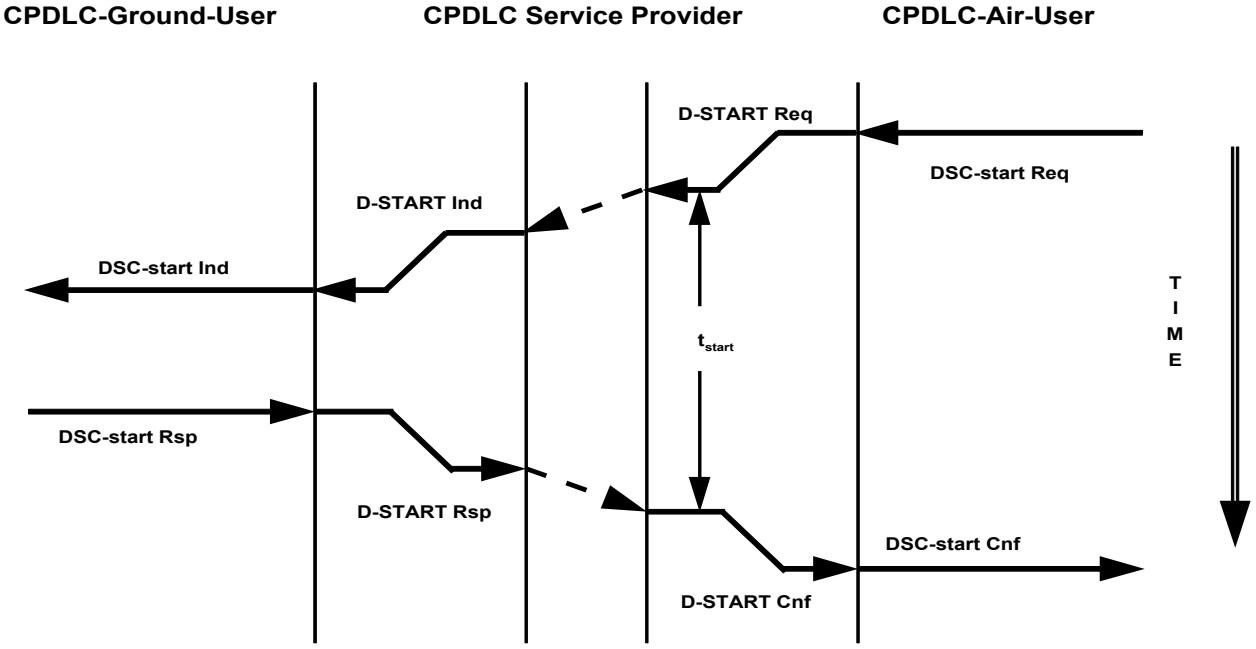


Figure 3.5-3. Sequence Diagram for DSC-start Service

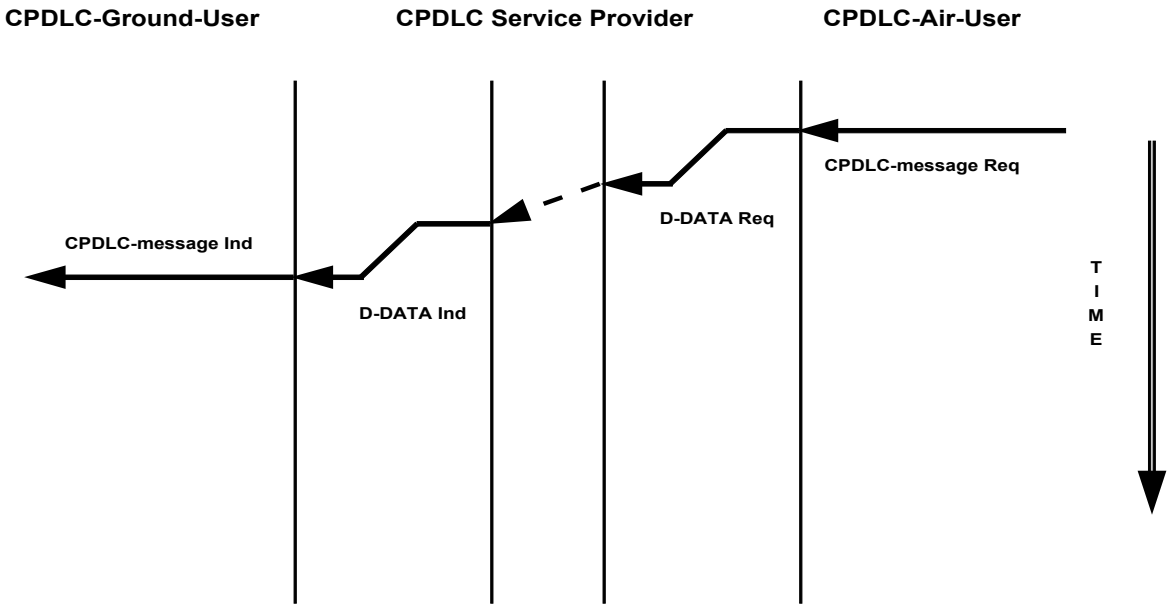


Figure 3.5-4. Sequence Diagram for CPDLC-message Service/Air Initiated

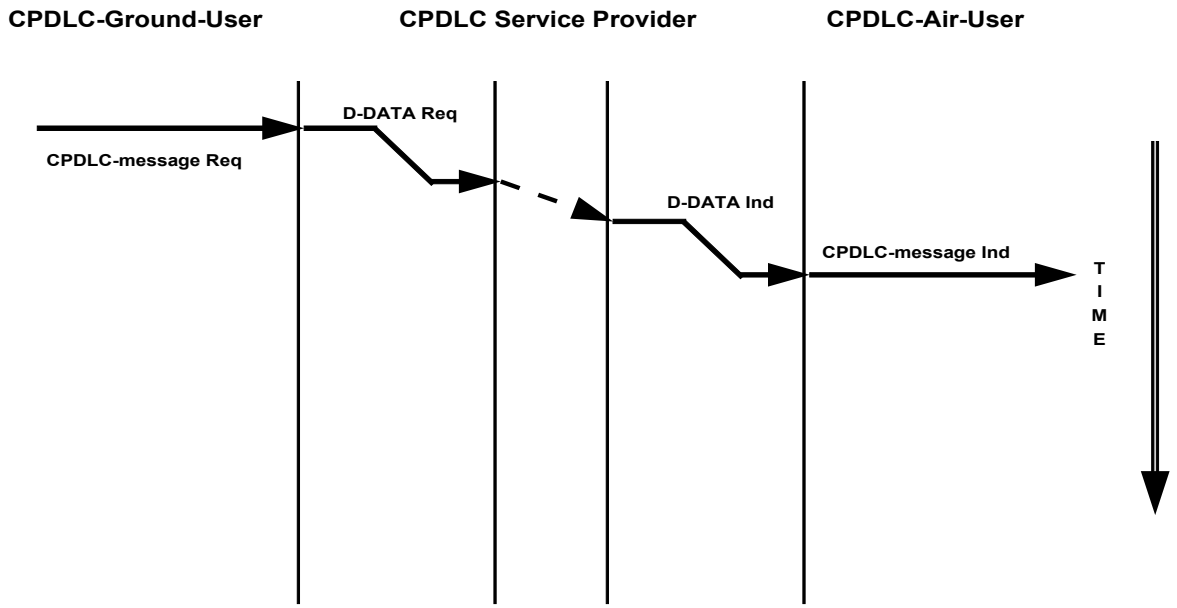


Figure 3.5-5. Sequence Diagram for CPDLC-message Service/Ground Initiated

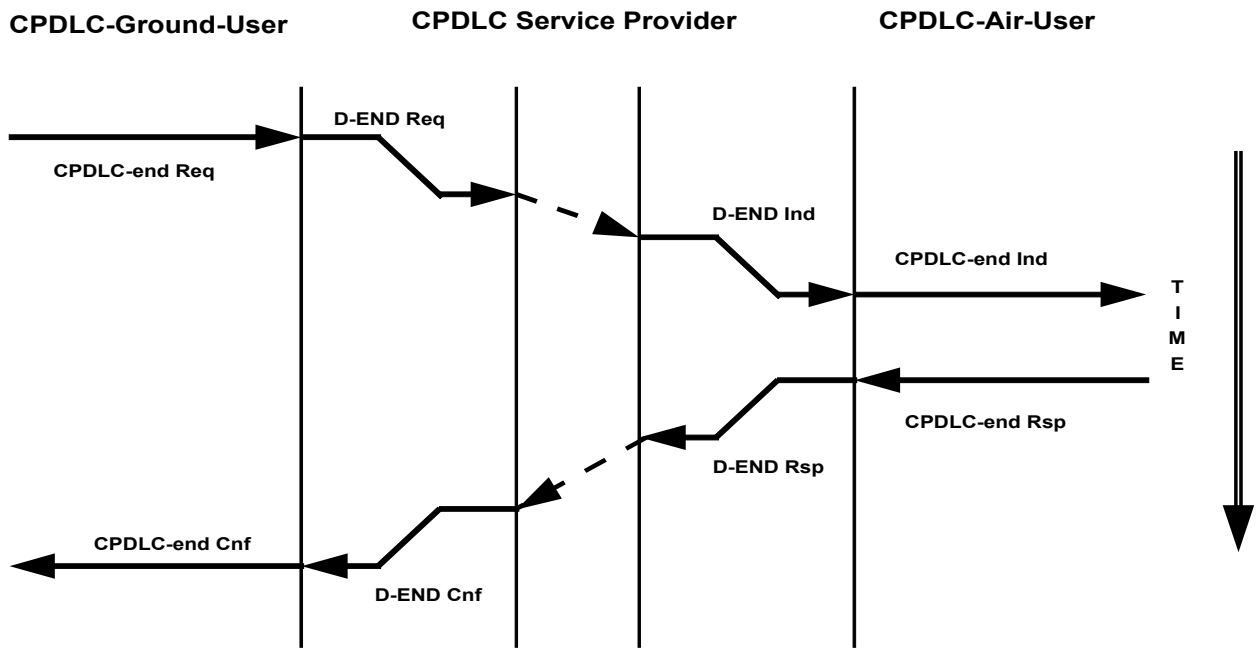


Figure 3.5-6. Sequence Diagram for CPDLC-end Service

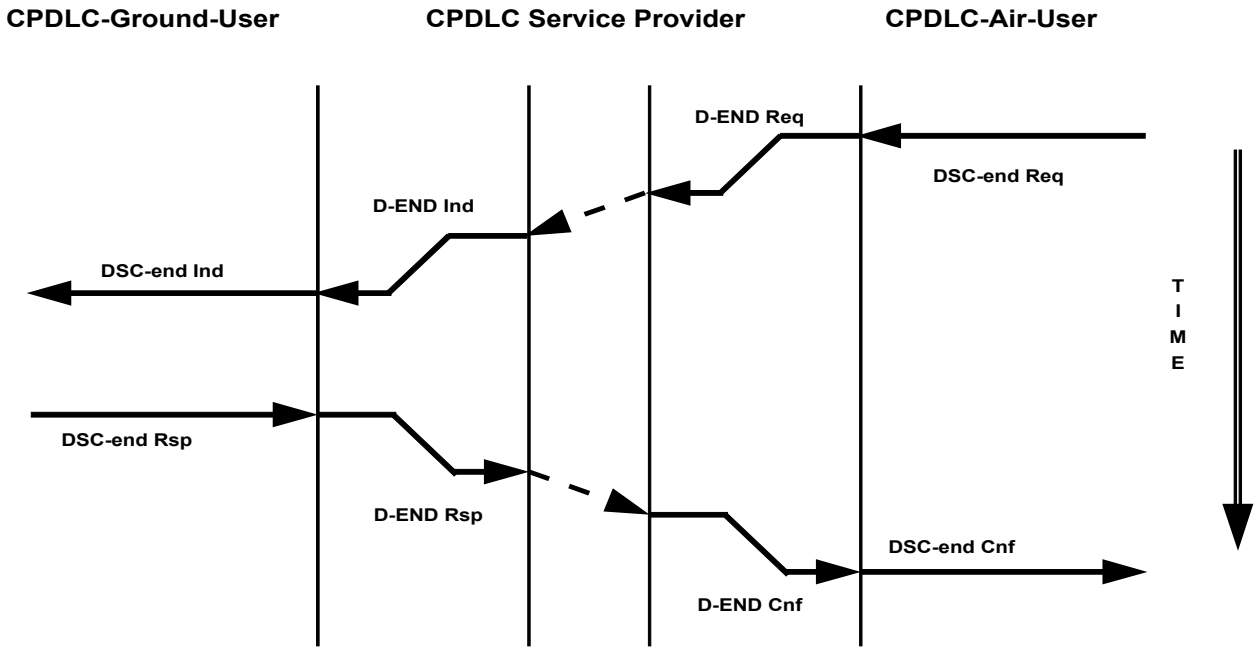


Figure 3.5-7. Sequence Diagram for DSC-end Service

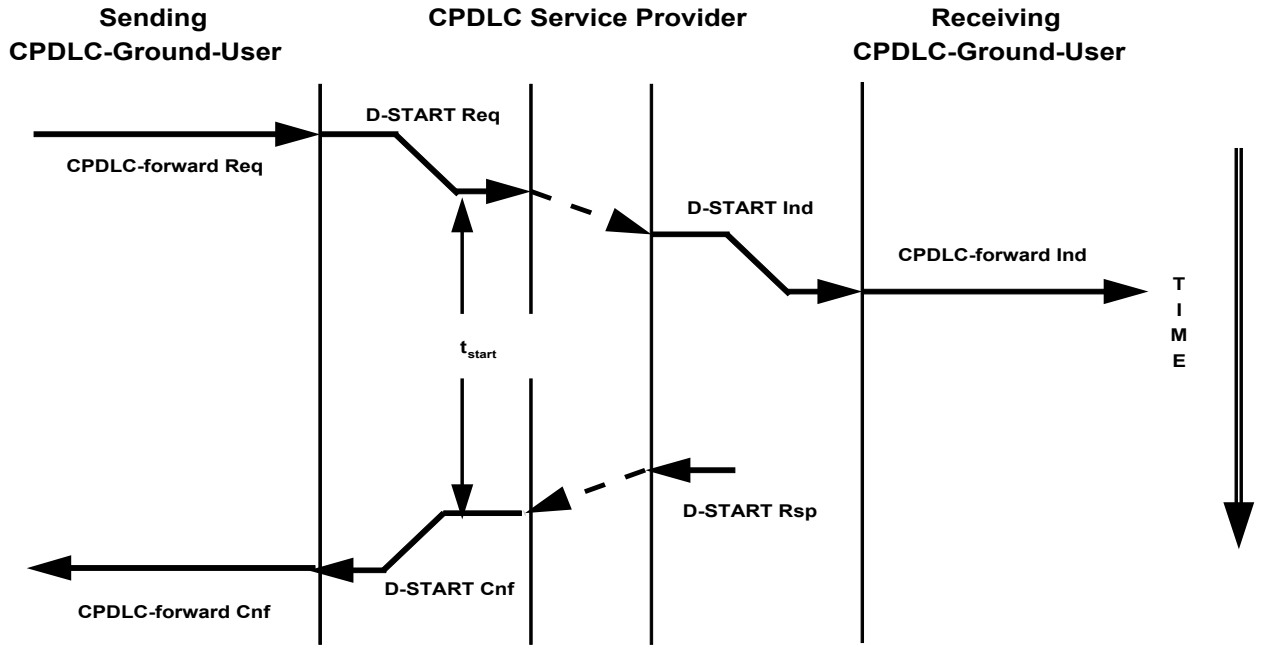


Figure 3.5-8. Sequence Diagram for CPDLC-forward Service/Ground Forwarding Supported, ASE Version Numbers the Same

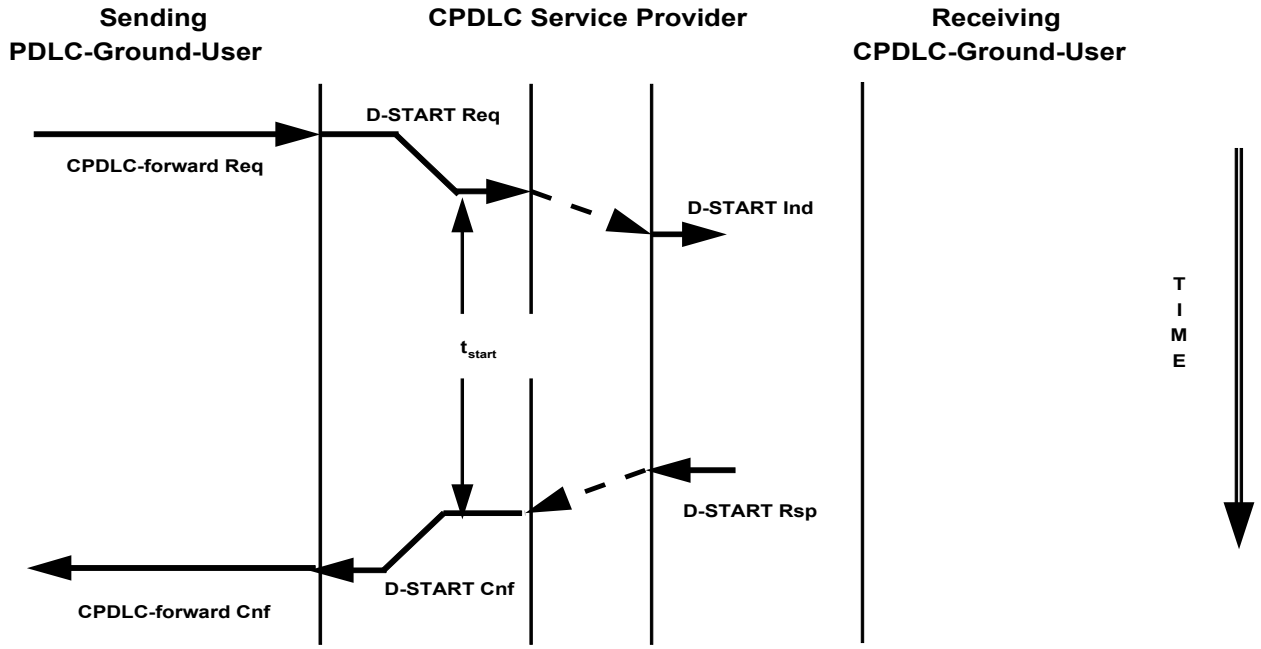


Figure 3.5-9. Sequence Diagram for CPDLC-forward Service/Ground Forwarding Not Supported, or Ground Forwarding Supported and ASE Version Numbers Not the Same

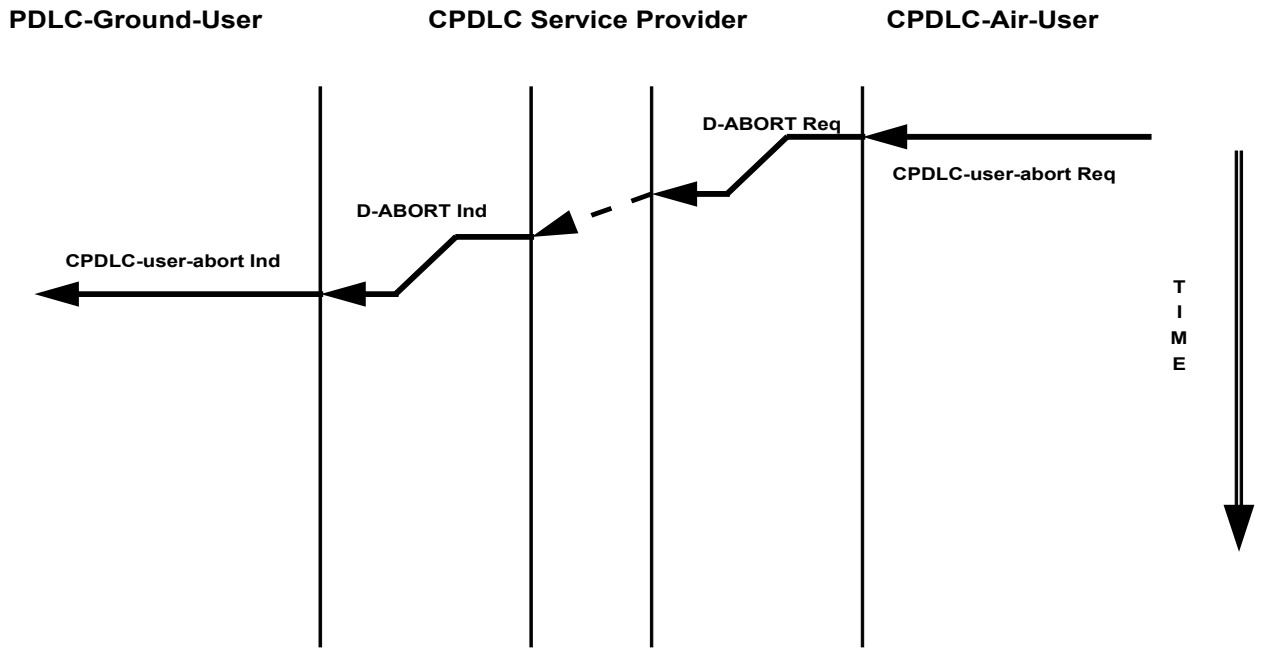


Figure 3.5-10. Sequence Diagram for CPDLC-user-abort Service/CPDLC-Air-User Initiated

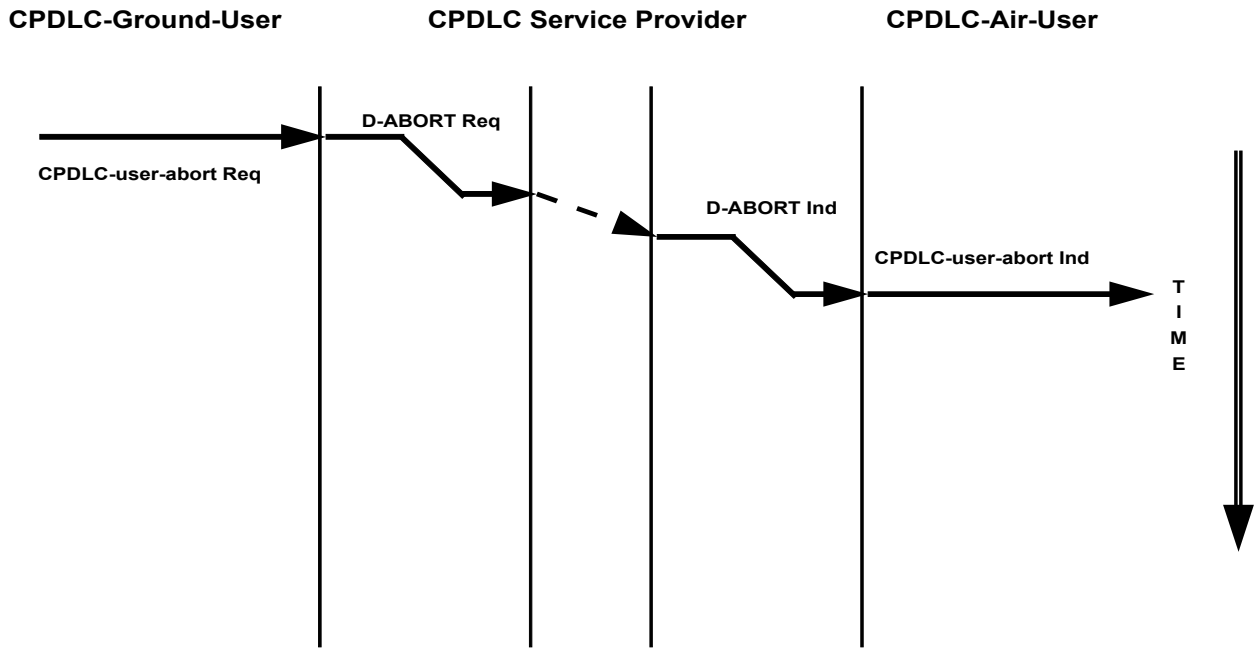


Figure 3.5-11. Sequence Diagram for CPDLC-user-abort Service/CPDLC-Ground-User Initiated

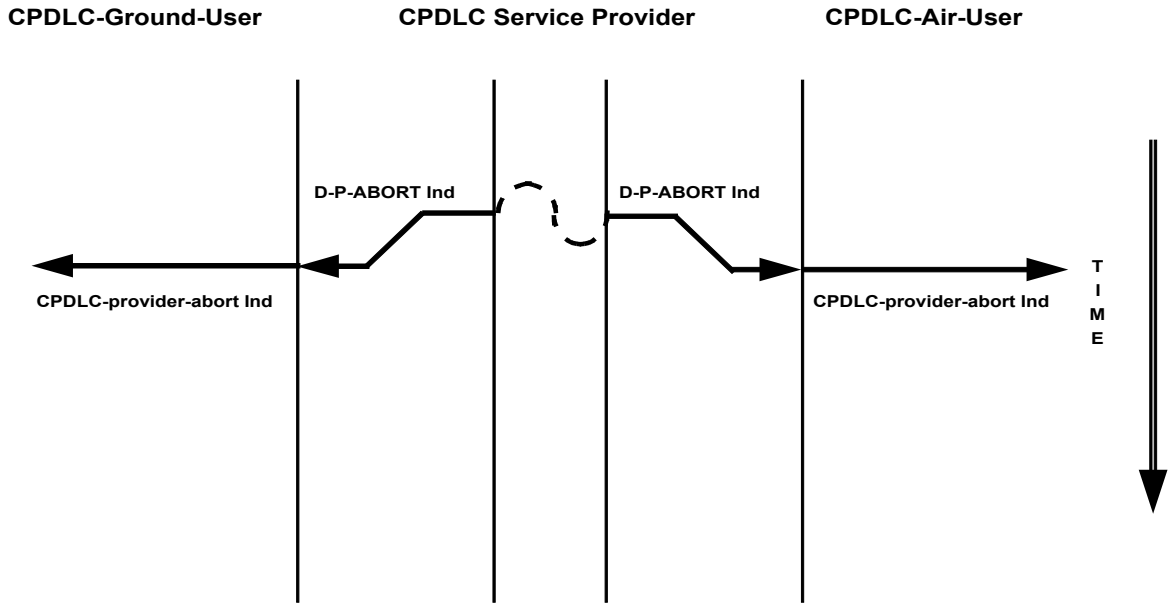


Figure 3.5-12. Sequence Diagram for CPDLC-provider-abort Service/Dialogue Service Abort

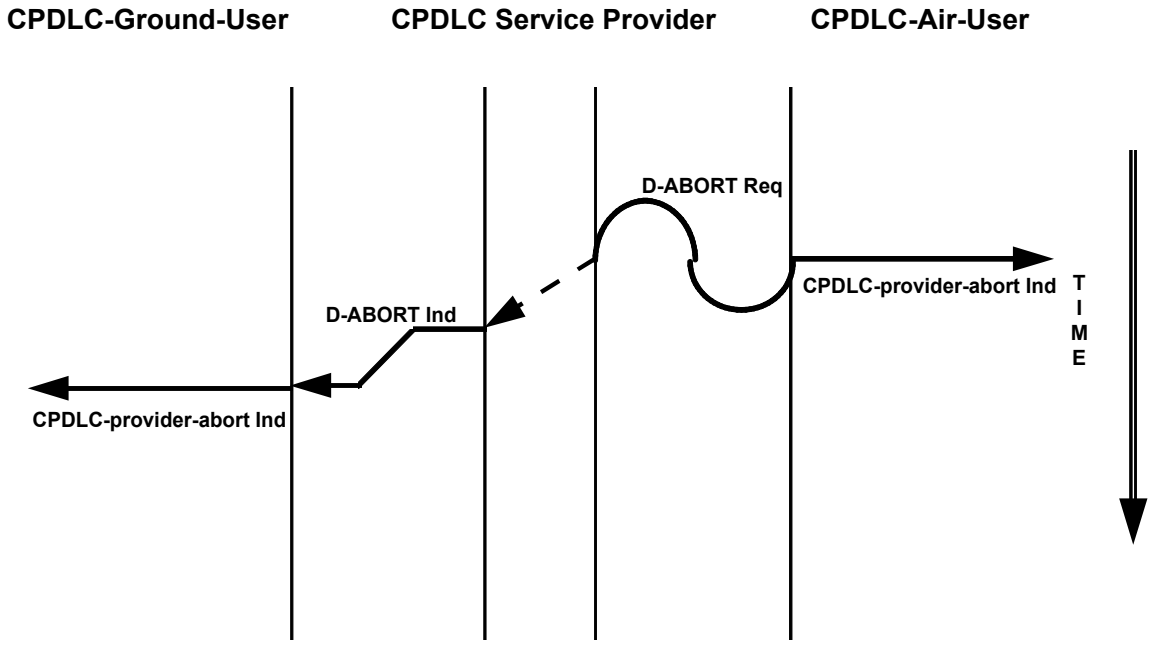


Figure 3.5-13. Sequence Diagram for CPDLC-provider-abort Service/CPDLC-Air-ASE Abort

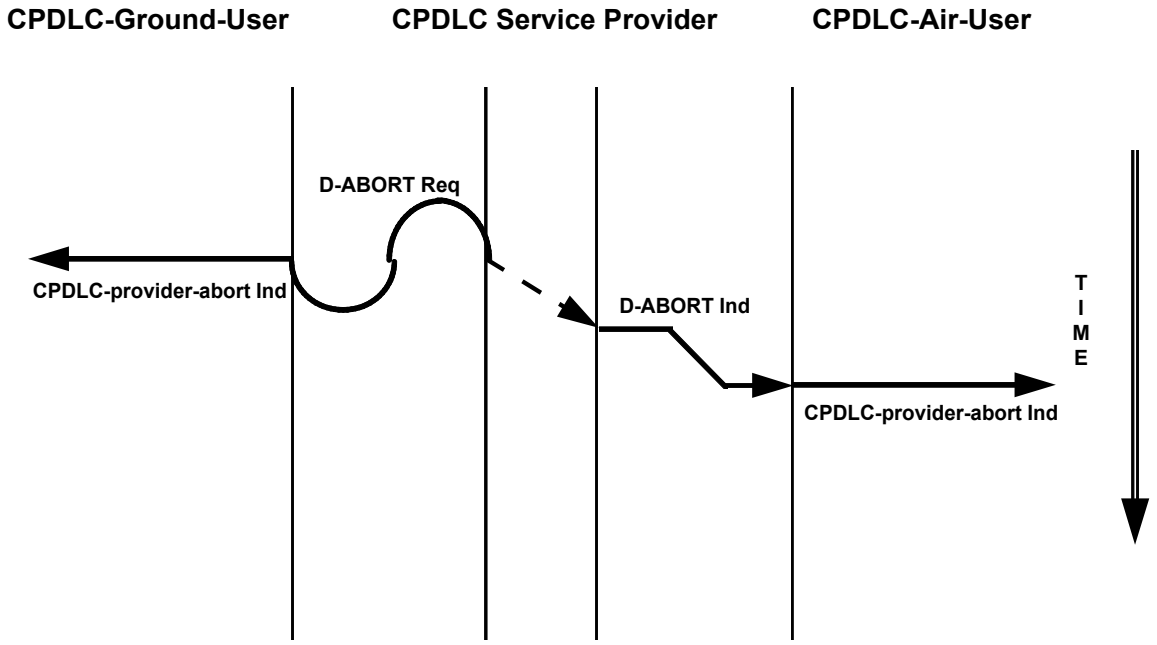


Figure 3.5-14. Sequence Diagram for CPDLC-provider-abort Service/ CPDLC-Ground-ASE Abort

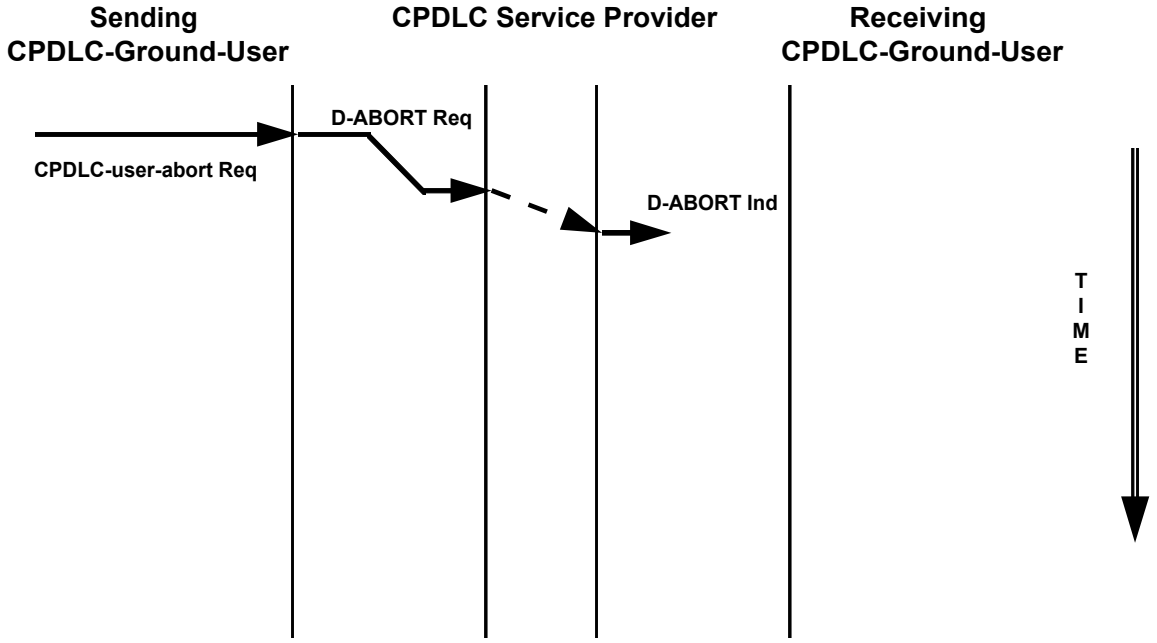


Figure 3.5-15. Sequence Diagram for CPDLC-user-abort Service/Sending CPDLC-Ground-User Initiated

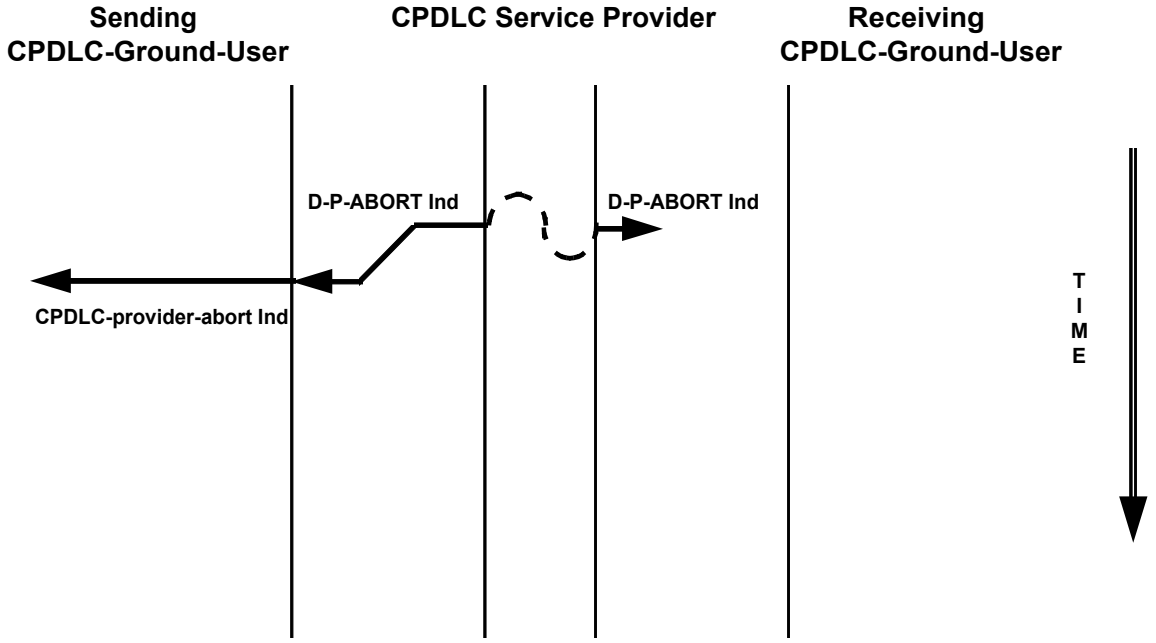


Figure 3.5-16. Sequence Diagram for CPDLC-provider-abort Service/Dialogue Service Abort

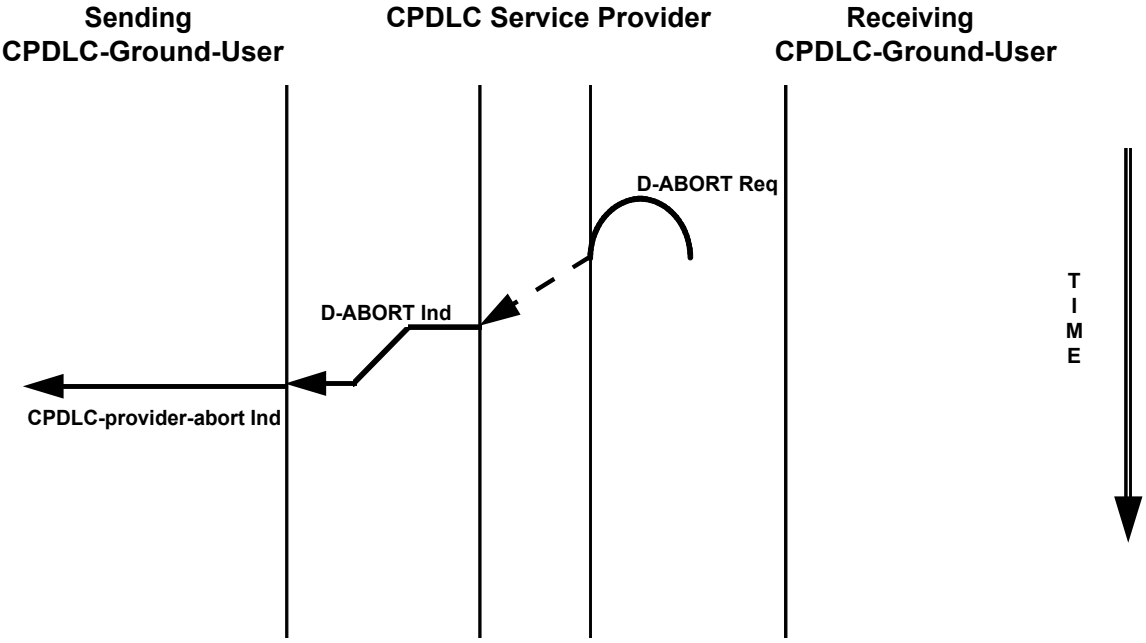


Figure 3.5-17. Sequence Diagram for CPDLC-provider-abort Service/Receiving CPDLC-Ground-ASE Abort

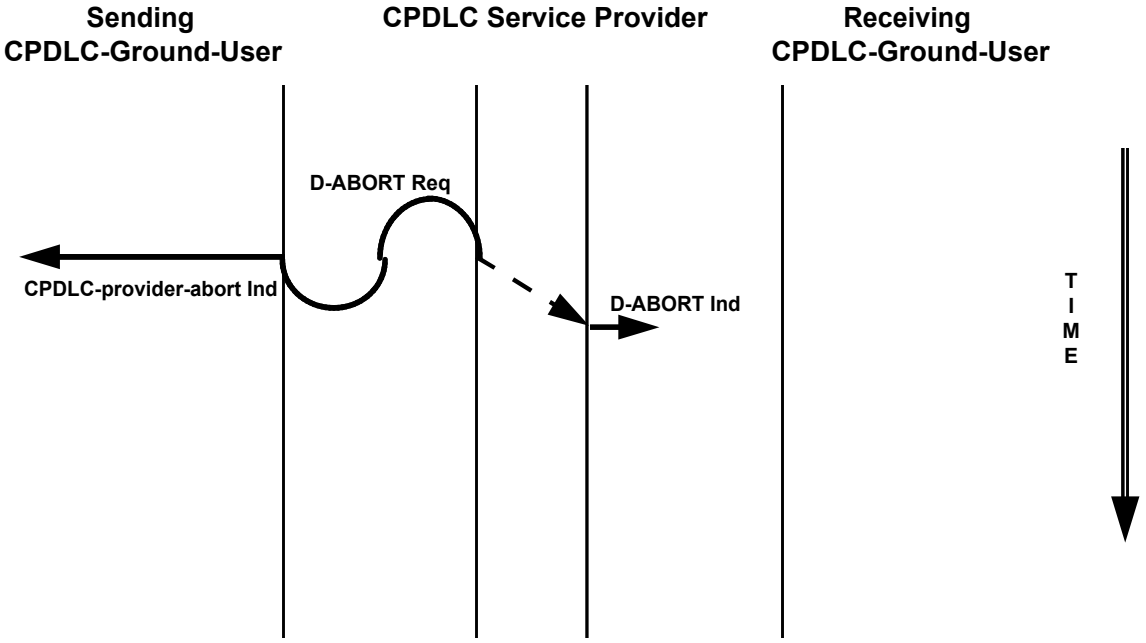


Figure 3.5-18. Sequence Diagram for CPDLC-provider-abort Service/Sending CPDLC-Ground-ASE Abort

3.5.2 CPDLC Service Provider Timers

3.5.2.1 A CPDLC-ASE shall be capable of detecting when a timer expires.

Note 1.— Table 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 3.5.4.1.

3.5.2.2 **Recommendation**.— *The timer values should be as indicated in Table 3.5-1.*

Table 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.

3.5.3 CPDLC-air-ASE Protocol Description

3.5.3.1 Introduction

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.

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 3.5.4.4 shall apply.

3.5.3.1.3 If a PDU is received that cannot be decoded, then exception handling procedures as described in 3.5.4.3 for invalid PDU shall apply.

3.5.3.1.4 If a PDU is not received when one is required, then exception handling as described 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.*

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.

3.5.3.1.6 On the initiation of a CPDLC-air-ASE, DSC shall be set to the abstract value "false".

3.5.3.2 D-START Indication

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 [ICUplinkMessage(startup)] 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", the D-START *QOS Routing Class* parameter has one of the abstract values specified in Table 3.6-1, and the D-START *Calling Peer ID* parameter is a valid four to eight character facility designation and the D-START *Security Requirements* parameter is consistent with the local security policy, 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) the GroundPDUs APDU-element as the CPDLC-start service *CPDLC/IC Data* parameter value, and

b) Enter the *START-IND* state.

3.5.3.3 D-START Confirmation

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 contains a GroundPDUs [ICUplinkMessage(send)] APDU and the D-START *Security Requirements* parameter value is the same as the one set for the D-START request, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:

1) the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Response CPDLC/IC Data* parameter value, and

2) the abstract value "accepted" as the CPDLC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

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 the D-START *User Data* parameter contains a GroundPDUs [ICUplinkMessage(send)] APDU and the D-START *Security Requirements* parameter value is the same as the one set for the D-START request, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:

1) the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Response CPDLC/IC Data* parameter value, and

2) the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and

c) Enter the *IDLE* state.

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 contains a GroundPDUs [ICUplinkMessage(send)] APDU and the D-START *Security Requirements* parameter value is the same as the one set for the D-START request, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke DSC-start service confirmation containing the following:

1) the APDU contained in the D-START *User Data* parameter as the DSC--start service *Response CPDLC/IC Data* parameter value, and

2) the abstract value "accepted" as the DSC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

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 the D-START *User Data* parameter contains a GroundPDUs [ICUplinkMessage(send)] APDU, the CPDLC-air-ASE shall:

a) Stop timer t_{start},

b) Invoke DSC-start service confirmation containing the following:

1) the APDU contained in the D-START *User Data* parameter as the DSC-start service *Response CPDLC/IC Data* 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.

3.5.3.4 D-DATA Indication

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 [ICUplinkMessage(send)] 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/IC Data* parameter value, and

b) Remain in the *DIALOGUE* state.

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 [ICUplinkMessage(send)] 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/IC Data* parameter value, and

b) Remain in the *END* state.

3.5.3.5 D-END Indication

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 the D-END *User Data* parameter contains a GroundPDUs [ICUplinkMessage(send)] 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-end service *CPDLC/IC Data* parameter value, and

b) Enter the *END* state.

3.5.3.6 D-END Confirmation

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 the D-END *User Data* parameter contains a GroundPDUs [ICUplinkMessage(send)] APDU, the CPDLC-air-ASE shall:

a) Invoke DSC-end service confirmation with:

1) the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC/IC Data* 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.

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 the D-END *User Data* parameter contains a GroundPDUs [ICUplinkMessage(send)] APDU, the CPDLC-air-ASE shall:

a) Invoke DSC-end service confirmation with:

1) the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC/IC Data* parameter value, and

2) the abstract value "rejected" as the CPDLC-end service *Result* parameter value,

b) Enter the *DIALOGUE* state.

3.5.3.7 CPDLC-start Service Request

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) the *CPDLC/IC Data* parameter as the ICDownlinkMessage,

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 CPDLC-start service *Security Required* parameter value if specified by the CPDLC-user else the abstract value "no security" as the D-START *Security Requirements* parameter value,

4) 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

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

c) Start timer t_{start}, and

d) Enter the *START-REQ* state.

3.5.3.8 CPDLC-start Service Response

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 DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ICDownlinkMessage APDU element based on the *Response CPDLC/IC Data* parameter,

b) Invoke D-START response with the following:

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

2) the abstract value received in the D-START indication *Security Requirements* parameter as the D-START *Security Requirements* parameter,

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

c) Enter the *DIALOGUE* state

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) Create an AircraftPDUs APDU with an ICDownlinkMessage APDU element based on the *Response CPDLC/IC Data* parameter,

b) Invoke D-START response with the following:

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

2) the abstract value received in the D-START indication *Security Requirements* parameter as the D-START *Security Requirements* parameter,

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

c) Enter the *IDLE* state.

3.5.3.9 DSC-start Service Request

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, and

2) the *CPDLC/IC Data* parameter as the ICDownlinkMessage,

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) the DSC-start service *Security Required* parameter value if specified by the CPDLC-air-user else the abstract value "no security" as the D-START *Security Requirements* parameter value,

4) 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

5) 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.

3.5.3.10 CPDLC-message Service Request

3.5.3.10.1 Upon receipt of a CPDLC-message service request and the *CPDLC/IC Data* parameter contains a CPDLC Message, if the CPDLC-air-ASE is in the *DIALOGUE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ICDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC/IC Data* parameter,

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

c) Remain in the *DIALOGUE* state.

3.5.10.2 Upon receipt of a CPDLC-message service request and the *CPDLC/IC Data* parameter contains a CPDLC Message, 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 ICDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC/IC Data* parameter,

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

c) Remain in the *END* state.

3.5.3.11 CPDLC-end Service Response

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 ICDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC/IC Data* parameter,

b) Invoke D-END response with the following:

1) 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.

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 ICDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC/IC Data* parameter,

b) Invoke D-END response with the following:

1) 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.

3.5.3.12 DSC-end Service Request

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 ICDownlinkMessage APDU-element based on the DSC-end service *CPDLC/IC Data* parameter,

b) Invoke D-END request with the APDU as the D-END *User Data* parameter value, and

c) Enter the *END* state.

3.5.3.13 CPDLC-user-abort Service Request

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.

3.5.3.14 D-ABORT Indication

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.

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.

3.5.3.15 D-P-ABORT Indication

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.

3.5.4 CPDLC-air-ASE Exception Handling

3.5.4.1 A Timer Expires

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.

3.5.4.2 Unrecoverable System Error

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.

3.5.4.3 Invalid PDU

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 the *User Data* parameter of a D-START confirmation with *Result* parameter set to the abstract value "accepted", 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.

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.

3.5.4.4 Protocol Error

3.5.4.4.1 If the *User Data* parameter of a D-START indication, D-START confirmation, 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 3.5.3, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [protocol-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 "protocol-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.

3.5.4.4.2 (deleted)

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 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 [protocol-error] 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 "protocol-error" 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.

3.5.4.4.4 Upon receipt of a Dialogue service primitive for which there are no instruction in 3.5.3 (i.e. the primitive was not expected or was expected under other conditions or with other parameter values), the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [protocol-error] APDU message element,

c) If a dialogue exists, 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 "protocol-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.

3.5.4.5 D-START Confirmation Result or Reject Source Not as Expected

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.

3.5.4.6 D-START Indication *Quality of Service* Not as Expected

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 3.6-1, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [invalid-QOS-parameter] 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 D-ABORT *User Data* parameter value,

d) If DSC has the abstract value "true", set DSC to the abstract value "false", and

e) Enter the *IDLE* state.

3.5.4.7 Expected PDU Missing

3.5.4.7.1 If the *User Data* parameter of a D-START indication or confirmation, a D-DATA indication, or a D-END indication or confirmation does not contain a PDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) Create an 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.

3.5.4.8 D-START Security Requirements Parameter Not as Expected

Note.— This section applies to the case when the D-START Security Requirements *parameter does not meet the local security policy.*

3.5.4.8.1 Upon receipt of a D-START indication with the *Security Requirements* parameter not consistent with the local security policy or upon receipt of a D-START confirmation with the *Security Requirements* parameter not equal to the value that was set in the D-START request, the CPDLC-air-ASE shall:

a) Stop all timers,

b) If the dialogue with the peer is still open:

1) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [communication-service-failure] 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,

c) If the CPDLC-air-user is active, invoke CPDLC-provider-abort service indication with the abstract value "communication-service-failure" as the CDPLC-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.

3.5.5 CPDLC-ground-ASE Protocol Description

3.5.5.1 Introduction

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.

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 3.5.6.4 shall apply.

3.5.5.1.3 If a PDU is received that cannot be decoded, then exception handling procedures as described in 3.5.6.3 for invalid PDU shall apply.

3.5.5.1.4 If a PDU is not received when one is required, then exception handling as described in 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.

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.

3.5.5.1.6 On the initiation of a CPDLC-ground-ASE, DSC shall be set to the abstract value "false".

3.5.5.2 D-START Indication

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 the ICAO 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 3.6-1 and the D-START *Security Requirements* parameter is consistent with the local security policy, 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, and

3) the AircraftPDUs APDU-element as the CPDLC-start service *CPDLC/IC Data* parameter value, and

b) Enter the *START-IND* state.

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 the ICAO 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 3.6-1, and the D-START *Security Requirements* parameter is consistent with the local security policy, 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) the AircraftPDUs APDU as the DSC-start service *CPDLC/IC Data* parameter value,

b) Set DSC to "true", and

c) Enter the *START-IND* state.

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 and the D-START *Security Requirements* parameter is consistent with the local security policy:

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.

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 and the D-START *Security*

Requirements parameter is consistent with the local security policy, 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.

3.5.5.3 D-START Confirmation

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 contains an AircraftPDUs [ICDownlinkMessage] APDU and the D-START *Security Requirements* parameter value is the same as the one set for the D-START request, the CPDLC-ground-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:

1) the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Response CPDLC/IC Data* parameter value, and

2) the abstract value "accepted" as the CPDLC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

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 the D-START *User Data* parameter contains an AircraftPDUs [ICDownlinkMessage] APDU and the D-START *Security Requirements* parameter value is the same as the one set for the D-START request, the CPDLC-ground-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:

1) the APDU contained in the D-START *User Data* parameter as the CPDLC-start service Response CPDLC/IC Data parameter value, and

2) the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and

c) Enter the *IDLE* state.

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 and the D-START *Security Requirements* parameter value is the same as the one set for the D-START request, the CPDLC-ground-ASE shall:

a) Stop timer t_{start},

b) invoke CPDLC-forward service confirmation with:

1) the D-START GroundPDUs APDU element as the CPDLC-forward service *Result* parameter value, and

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

c) enter the *IDLE* state.

3.5.5.4 D-DATA Indication

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 [ICDownlinkMessage] 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/IC Data* parameter value, and

b) Remain in the *DIALOGUE* state.

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 [ICDownlinkMessage] 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/IC Data* parameter value, and

b) Remain in the *END* state.

3.5.5.5 D-END Indication

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 the D-END *User Data* parameter contains an AircraftPDUs [ICDownlinkMessage] 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/IC Data* parameter value, and

b) Enter the *END* state.

3.5.5.6 D-END Confirmation

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 the D-END *User Data* parameter contains an AircraftPDUs [ICDownlinkMessage] 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/IC Data* parameter value, and

2) The abstract value "accepted" as the CPDLC-end service *Result* parameter value, and

b) Enter the *IDLE* state.

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 the D-END *User Data* parameter contains an AircraftPDUs [ICDownlinkMessage] 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/IC Data* parameter value, and

2) The abstract value "rejected" as the CPDLC-end service *Result* parameter value, and

b) Enter the *DIALOGUE* state.

3.5.5.7 CPDLC-start Service Request

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(startup) APDU element containing the *CPDLC/IC Data* parameter 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 CPDLC-start service *Security Required* parameter value if specified by the CPDLC-user else the abstract value "no security" as the D-START *Security Requirements* parameter value,

4) 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

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

c) Start timer t_{start}, and

d) Enter the *START-REQ* state.

3.5.5.8 CPDLC-start Service Response

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 DSC has the abstract value "false", the CPDLC-ground-ASE shall:

a) create a GroundPDUs APDU with a ICUplinkMessage(send) APDU element based on the *Response CPDLC/IC Data* parameter,

b) Invoke D-START response with the following:

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

2) the abstract value received in the D-START indication *Security Requirements* parameter as the D-START *Security Requirements* parameter,

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

c) Enter the *DIALOGUE* state.

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) create a GroundPDUs APDU with a ICUplinkMessage(send) APDU element based on the *Response CPDLC/IC Data* parameter,

b) Invoke D-START response with the following:

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

2) the abstract value received in the D-START indication *Security Requirements* parameter as the D-START *Security Requirements* parameter, and

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

c) Enter the *IDLE* state.

3.5.5.9 DSC-start Service Response

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) create a GroundPDUs APDU with an ICUplinkMessage(send) APDU element based on the *Response CPDLC/IC Data* parameter,

b) Invoke D-START response with the following:

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

2) the abstract value received in the D-START indication *Security Requirements* parameter as the D-START *Security Requirements* parameter, and

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

c) Enter the *DIALOGUE* state.

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) create a GroundPDUs APDU with an ICUplinkMessage(send) APDU element based on the *Response CPDLC/IC Data* parameter,

b) Invoke D-START response with the following:

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

2) the abstract value received in the D-START indication *Security Requirements* parameter as the D-START *Security Requirements* parameter, and

3) 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.

3.5.5.10 CPDLC-message Service Request

3.5.5.10.1 Upon receipt of a CPDLC-message service request and the *CPDLC/IC Data* parameter contains a CPDLC Message, if the CPDLC-ground-ASE is in the *DIALOGUE* state, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ICUplinkMessage(send) APDU element based on the CPDLC-message service *CPDLC/IC Data* parameter,

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

c) Remain in the *DIALOGUE* state.

3.5.5.10.2 Upon receipt of a CPDLC-message service request and the *CPDLC/IC Data* parameter contains a CPDLC Message, 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 ICUplinkMessage(send) APDU element based on the CPDLC-message service *CPDLC/IC Data* parameter,

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

c) Remain in the *END* state.

3.5.5.11 CPDLC-end Service Request

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 ICUplinkMessage(send) APDU element based on the CPDLC-end service *CPDLC/IC Data* parameter,

b) Invoke D-END request with the APDU as the D-END *User Data* parameter value, and,

c) Enter the *END* state.

3.5.5.12 DSC-end Service Response

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 ICUplinkMessage(send) APDU element based on the DSC-end service *CPDLC/IC Data* parameter,

b) Invoke D-END response with the following:

1) the APDU as the D-END *User Data* parameter; 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.

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 ICUplinkMessage(send) APDU element based on the DSC-end service *CPDLC/IC Data* parameter,

b) Invoke D-END response with the following:

1) the APDU as the D-END *User Data* parameter; and

2) the abstract value "rejected" as the D-END *Result* parameter value, and

c) Enter the *DIALOGUE* state.

3.5.5.13 CPDLC-forward Service Request

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 CPDLC-forward service *Security Required* parameter value if specified by the CPDLC-user else the abstract value "no security" as the D-START *Security Requirements* parameter value,

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

i) if provided, the CPDLC-forward 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

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

c) Start timer t_{start}, and

d) Enter the *FORWARD* state.

3.5.5.14 CPDLC-user-abort Service Request

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.

3.5.5.15 D-ABORT Indication

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.

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.

3.5.5.16 D-P-ABORT Indication

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.

3.5.6 CPDLC-ground-ASE Exception Handling

3.5.6.1 A Timer Expires

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.

3.5.6.2 Unrecoverable System Error

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.

3.5.6.3 Invalid PDU

3.5.6.3.1 If the *User Data* parameter of a D-END confirmation with *Result* parameter set to the abstract value "rejected", or the *User* Data parameter of a D-START confirmation with *Result* parameter set to the abstract value "accepted", 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.

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.

3.5.6.4 Protocol Error

3.5.6.4.1 If the *User Data* parameter of a D-START indication, a D-START confirmation, a 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 3.5.5, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [protocol-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 "protocol-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.

3.5.6.4.2 (deleted)

3.5.6.4.2.1 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 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 [protocol-error] 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 "protocol-error" 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.

3.5.6.4.2.2 Upon receipt of a Dialogue service primitive for which there are no instruction in 3.5.3 (i.e. the primitive was not expected or was expected under other conditions or with other parameter values), the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [protocol-error] APDU message element,

c) If a dialogue exists, 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 "protocol-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.

3.5.6.5 D-START Confirmation Result or Reject Source Not as Expected

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.

3.5.6.6 D-START Indication Quality of Service Not as Expected

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 3.6-1, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [invalid-QOS-parameter] 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 DSC has the abstract value "true", set DSC to the abstract value "false", and

e) Enter the *IDLE* state.

3.5.6.7 Expected PDU Missing

3.5.6.7.1 If the *User Data* parameter of a D-START indication or confirmation, a D-DATA indication, or a D-END indication or confirmation 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.

3.5.6.8 D-START Security Requirements Parameter Not as Expected

Note.— This section applies to the case when the D-START Security Requirements *parameter does not meet the local security policy.*

3.5.6.8.1 Upon receipt of a D-START indication with the *Security Requirements* parameter not consistent with the local security policy or upon receipt of a D-START confirmation with the *Security Requirements* parameter not equal to the value that was set in the D-START request, the CPDLC-ground-ASE shall:

a) Stop all timers,

b) If the dialogue with the peer is still open:

1) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [communication-service-failure] 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,

c) If the CPDLC-ground-user is active, invoke CPDLC-provider-abort service indication with the abstract value "communication-service-failure" as the CDPLC-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.

3.5.7 CPDLC ASE State Tables

3.5.7.1 Priority

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 3.5-2. CPDLC-air-ASE State Table

STATE → EVENT ↓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
Dialogue Service Events					
D-START Indication APDU = ICUplinkMessage (startup)	● CPDLC-start indication →*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "false" APDU = ICUplinkMessage (send)	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" APDU = ICUplinkMessage (send)	cannot occur	●Stop timer t_{start} ● CPDLC-start confirmation →*IDLE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "true" APDU = ICUplinkMessage (send)	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" APDU = ICUplinkMessage (send)	cannot occur	●Stop timer t_{start} ● DSC-start confirmation ●Set DSC = "false" →*IDLE*	cannot occur	cannot occur	cannot occur
D-DATA Indication APDU = ICUplinkMessage (send)	cannot occur	cannot occur	cannot occur	●*CPDLC*-message indication →*DIALOGUE*	●if DSC="true" ● CPDLC-message indication →*END* ●else cannot occur

STATE → EVENT ↓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
D-END Indication: DSC="false" APDU = ICUplinkMessage (send)	cannot occur	cannot occur	cannot occur	● CPDLC-end indication →*END*	cannot occur
D-END Confirmation: DSC="true" *Result* "accepted" APDU = ICUplinkMessage (send)	cannot occur	cannot occur	cannot occur	cannot occur	● DSC-end confirmation ●Set DSC "false" →*IDLE*
D-END Confirmation: DSC="true", *Result* "rejected" APDU = ICUplinkMessage (send)	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*

STATE → EVENT ↓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
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*
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 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-DATA Indication	cannot occur	cannot occur	cannot occur	●CPDLC-message indication →*DIALOGUE*	●if DSC = "false ●CPDLC-message indication →*END* ●else not permitted	cannot occur

STATE → **EVENT ↓**	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
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
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

STATE → **EVENT ↓**	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
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*
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*

STATE → **EVENT ↓**	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
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*

— — — — — — — —

Chapter 3

(Sections 3.6, 3.7 and 3.8)

(See mapping table for conversion of current paragraph numbers of Doc 9705 – 3rd edition into paragraph numbers of Doc 9880)

3.6 COMMUNICATION REQUIREMENTS

3.6.1 Encoding Rules

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 CPDLC message structure and content specified in 3.4.

Note.— When CPDLC APDUs are treated as bit-oriented values that are not padded to an integer number of octets, the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

3.6.2 Dialogue Service Requirements

3.6.2.1 Primitive Requirements

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 3.5, the CPDLC-ground-ASE and the CPDLC-air-ASE shall exhibit external behaviour consistent with the dialogue service, as described in Doc 9705, Sub-volume IV, section 4.2 having been implemented and its primitives invoked.

3.6.2.2 Quality-of-Service Requirements

3.6.2.2.1 The application service priority for CPDLC shall have the abstract value of “high priority flight safety messages”.

3.6.2.2.2 The RER Quality of Service Parameter of the D-START shall be set to the abstract value of “low”.

3.6.2.2.3 [PDR98120009/11]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 3.6-1.

Table 3.6-1. Mapping Between Class of Communication and Routing Class Abstract Values

Class of Communication Abstract Value	Routing Class Abstract Value
No Preference	No Traffic Type Policy Preference
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 Doc 9705, Sub-volume I, paragraph 1.3.

3.6.2.3 ATN Security Requirements

3.6.2.3.1 The *Security Requirements* parameter of the D-START shall be set to either "secured exchange" or "no security" for the CPDLC-start, DSC-start and CPDLC-forward services.

— — — — — — — —

3.7 CPDLC USER REQUIREMENTS

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.— "CPDLC/IC Data" is:

a) *what the CPDLC-user provides to the CPDLC-service as the CPDLC/IC Data or Response CPDLC/IC Data parameter, when invoking a CPDLC-service request or response primitive,*

b) or what the CPDLC-user receives in the same parameters from the CPDLC-service indication or confirmation primitives.

Note 4.— A "CPDLC Message" is an uplink or downlink message exchanged between CPDLC-users as part of the CPDLC/IC Data.

Note 5.— 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/IC Data value has been provided by the CPDLC-service.

Note 6.— A CPDLC-air-user uses the contents of an uplink CPDLC/IC Data item to verify that it has been delivered to the correct destination, as identified by the combination of Flight ID and ICAO 24-bit Aircraft Address, and is from the appropriate Data Authority, and that it was encoded and decoded by the CPDLC-users using the same CPDLC abstract syntax version.

Note 7.— A CPDLC-ground-user uses the contents of a downlink CPDLC/IC Data item to verify that it has been delivered to the correct destination and is from the expected aircraft as identified by the combination of Flight ID and ICAO 24-bit Aircraft Address, and that it was encoded and decoded by the CPDLC-users using the same CPDLC abstract syntax version.

Note 8.— The verification of correct delivery is performed by the CPDLC-user and not by the CPDLC-ASE. This avoids placing any reliance on the communications protocols or their implementation for a proof of correct delivery or a proof that data integrity has been maintained.

Note 9.— Use of an Integrity Check generation/validation algorithm other than the ATN Message Checksum specified in chapter 6 of this manual is out of scope of this specification and requires prior agreement by both CPDLC air and ground users.

3.7.1.1 General CPDLC-service requirements

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.

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.

Note 3.— When the CPDLC-user requires secure or unsecure services, the user sets the Security Required parameter as appropriate.

3.7.2 CPDLC/IC Data Generation Requirements

Note.— When a CPDLC-user requires to send a CPDLC Message within the CPDLC/IC Data parameter of a CPDLC Start, a DSC Start, a CPDLC Message, a CPDLC End, or a DSC End primitive, it is implied that the CPDLC Message is encoded within an ICUplinkMessage or ICDownlinkMessage as specified in this section.

3.7.2.1 Downlink CPDLC/IC Data

3.7.2.1.1 The CPDLC-air-user shall compose a downlink CPDLC/IC Data value from:

a) an optional identifier for the algorithm used to compute the Integrity Check,

b) an optional downlink CPDLC Message, and

c) an Integrity Check,

Note 1.— The algorithm identifier identifies both the algorithm used to generate the checksum and the data over which the checksum is generated. In the default case, the algorithm used to compute the checksum is the default ATN Message Checksum generation algorithm, and the data over which the checksum is generated is the PseudoCPDLCMessage, specified in 3.7.3.2.

Note 2.— The Integrity Check can be generated even when a CPDLC Message is not present. With the default Integrity Check, this still permits the sender and receiver to be verified.

3.7.2.1.2 Unless it is has been agreed by means outside of this specification that an alternative

Integrity Check generation algorithm is used, the CPDLC-air-user shall compute the Integrity Check using the default *Integrity Check* generation algorithm as specified in Chapter 6.

3.7.2.1.3 The default Integrity Check generation algorithm shall be the computation of the Integrity Check using the default ATN Message Checksum specified in Chapter 6 of this manual over the PseudoCPDLCMessage specified in 3.7.3.2.

3.7.2.1.4 When a downlink CPDLC Message is provided in a *CPDLC/IC Data* or *Response CPDLC/IC Data* parameter, it shall be created by the CPDLC-air-user from an ATCDownlinkMessage as defined in 3.4.3, and encoded using the Basic Unaligned PER variant.

Note.— When the CPDLC Message is treated as bit-oriented values that are not padded to in integral number of octets, the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

3.7.2.1.5 When the value of the Integrity Check is computed using the default *Integrity Check* generation algorithm specified in Chapter 6 of this manual, the CPDLC-air-user shall omit the algorithm identifier on the first downlink CPDLC/IC Data value, or set it to the algorithm identifier "atn-default-checksum".

3.7.2.1.6 When an Integrity Check generation algorithm other than the default is used, the CPDLC-air-user shall set the algorithm identifier on the first downlink CPDLC/IC Data value to the value of the relative OID that identifies the Integrity Check generation algorithm used as specified in Chapter 6.

3.7.2.1.7 In subsequent downlink CPDLC/IC Data values, the CPDLC-air-user shall omit the algorithm identifier.

3.7.2.2 Uplink CPDLC/IC Data

3.7.2.2.1 The CPDLC-ground-user shall compose an uplink CPDLC/IC Data value from:

a) an optional identifier for the algorithm used to compute the Integrity Check,

b) an optional uplink CPDLC Message and

c) an Integrity Check.

Note.— The Integrity Check can be generated even when a CPDLC Message is not present. With the default Integrity Check, this still permits the sender and receiver to be verified.

3.7.2.2.2 Unless it is has been agreed by means outside of this specification that an alternative Integrity Check algorithm is used, the CPDLC-ground-user shall compute the Integrity Check using the default *Integrity Check generation algorithm* specified in chapter 6 of this manual.

3.7.2.2.3 When an uplink CPDLC Message is provided in a *CPDLC/IC Data* or *Response CPDLC/IC Data* parameter, it shall be created by the CPDLC-ground-user from an ATCUplinkMessage as defined in

3.4.3, and encoded using the Basic Unaligned PER variant.

Note.— When the CPDLC Message is treated as bit-oriented values that are not padded to in integral number of octets, the length determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

3.7.2.2.4 When the value of the Integrity Check is computed using the default Integrity Check generation algorithm generation algorithm specified in chapter 6 of this manual, the CPDLC-ground-user shall omit the algorithm identifier on the first uplink CPDLC/IC Data value, or set it to the algorithm identifier "atn-default-checksum".

3.7.2.2.5 When an Integrity Check generation algorithm other than the default is used, the CPDLC-ground-user shall set the algorithm identifier on the first uplink CPDLC/IC Data value to the value of the relative OID that identifies the Integrity Check generation algorithm used.

3.7.2.2.6 In subsequent uplink CPDLC/IC Data values, the CPDLC-ground-user shall omit the algorithm identifier.

3.7.3 The Integrity Check

3.7.3.1 The message to be used by the sending CPDLC-user to generate the Integrity Check shall consist of the unaligned basic PER encoding of the ASN.1 PseudoCPDLCMessage data type, created from:

a) the aircraft Flight Identification assigned to the CPDLC-air-user, expressed in canonical form as defined in 3.7.3.4,

b) the ICAO 24-bit aircraft address assigned to the CPDLC-air-user,

c) the Ground Facility Designator that identifies the CPDLC-ground-user for the CPDLC or DSC dialogue, being the same as that used for the *Called Peer Identifier* or *Calling Peer Identifier* parameter of the CPDLC-start service or DSC-start service used to initiate the CPDLC or DSC dialogue,

d) the abstract syntax identifier of the CPDLC Message used for the current CPDLC or DSC dialogue, and

e) the unaligned basic PER encoded value of the ATCUplinkMessage or ATCDownlinkMessage, as appropriate, if any such message is present.

Note 1.— The Ground Facility Designator is used as the called peer identifier for air initiated CPDLC or DSC dialogues and as the calling peer identifier for ground initiated CPDLC dialogues.

Note 2.— The PseudoCPDLCMessage is created solely for the purposes of Integrity Check computation and is never exchanged over a CPDLC or DSC dialogue.

The PseudoCPDLCMessage data type shall comply with the ASN.1 module ATCMessageIntegrityCheckVersion1, as defined in this section.

```
ATCMessageIntegrityCheckVersion1 DEFINITIONS ::=

BEGIN

IMPORTS
        AircraftFlightIdentification,
        AircraftAddress,
        FacilityDesignation FROM CPDLCMessageSetVersion1;

PseudoCPDLCMessage ::= SEQUENCE
        {
                flightID                        AircraftFlightIdentification,
                aircraftAddress                 AircraftAddress,
                facilityDesignator              FacilityDesignation,
                cPDLCMessageAbstractSyntax      OBJECT IDENTIFIER,
                embeddedMessage                 EncodedCPDLCMessage OPTIONAL
        }

EncodedCPDLCMessage ::= BIT STRING

END
```

3.7.3.2 Aircraft Flight Identification

Note.— The Integrity Check can only be correctly verified, if both CPDLC ground and air users use the same Aircraft Flight Identification and expressed in exactly the same syntax. Either *the Aircraft Registration (Tail Number) or the Flight Identification can be used but, in either case, the form used must be identical to that given in the associated Flight Plan (see ICAO Doc 4444, App 2).*

3.7.3.2.1 The Aircraft Flight Identification (FlightID) shall be the Aircraft Identification as given in Field 7 of the filed Flight Plan.

3.7.3.2.2 When the Aircraft Registration Marking is used as the FlightID, the FlightID shall be the registration marking of the aircraft expressed as a character string (maximum seven characters) using upper case IA5 alphabetic and numeric characters only and without any spaces or hyphens.

3.7.3.2.3 When the Flight Identification is used as the FlightID, the FlightID shall be a character string (maximum seven characters) using upper case IA5 alphabetic and numeric characters only and without any spaces or hyphens, and formed from the three letter ICAO Designator for the Aircraft Operating Agency (see ICAO Doc 8585) followed by the flight number expressed as a numeric string with no leading zeroes.

Note.— Although the ASN.1 definition of the AircraftFlightIdentification permits the Flight ID to be expressed as a character string of up to eight characters, ICAO Doc 4444 places a limit of seven characters on the Aircraft Flight Identification in the Flight Plan.

3.7.4 CPDLC Message Generation Requirements

Note 1.— A Response CPDLC/IC Data 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 CPDLC/IC Datas 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 CPDLC/IC Datas.

Note 3.— A closure Response CPDLC/IC Data 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.

Note 4.— The CPDLC Message specified in this section is the uplink or downlink message exchanged between the peer CPDLC-users. It is exchanged in PER-encoded form as the embeddedMessage element of a CPDLC/IC Data value.

3.7.4.1 Message Composition

3.7.4.1.1 A CPDLC Message shall be composed of a message header, and from one to five message elements.

3.7.4.1.2 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 ACKNOWLEDGEMENT requirement (optional).

3.7.4.1.3 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.

3.7.4.1.4 A message element shall consist of a message element identifier, data as indicated by the specified message element, and associated message element attributes.

3.7.4.1.5 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 CPDLC/IC Data,

c) date and time,

d) a logical ACKNOWLEDGEMENT indication, if required,

e) from one to five message element identifiers, and

f) data as required for each message element identification included.

3.7.4.1.6 For each CPDLC Message the CPDLC-user sends ground/ground it shall provide the following information:

a) the aircraft flight 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.

3.7.4.2 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 with 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 CPDLC ground system and those message identification numbers assigned and managed by the aircraft.

3.7.4.2.1 The message identification number provided by the CPDLC-user shall be different from any other message identification number currently in use.

3.7.4.2.2 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.

3.7.4.2.3 If a CPDLC or DSC dialogue is terminated, all message identification numbers pertaining to that dialogue shall be considered available.

3.7.4.3 Message Elements That Cannot Be Combined With Other Message Elements in a Message

3.7.4.3.1 The LOGICAL ACKNOWLEDGEMENT message element (uplink message element 227 and downlink message element 100) shall be sent only as a single message element CPDLC Message.

3.7.4.3.2 The NEXT DATA AUTHORITY [facility] message element (uplink message element 160) shall be sent only as a single message element CPDLC Message.

3.7.4.4 Restriction on Route Clearance Variable Message Elements

3.7.4.4.1 A CPDLC Message shall contain no more than two message elements with the [routeClearance] variable.

3.7.5 CPDLC/IC Data Receipt Requirements

3.7.5.1 Downlink CPDLC/IC Data

3.7.5.1.1 On receipt of a downlink CPDLC/IC Data value, the Integrity CheckIntegrity Check verification procedure shall be performed before any other CPDLC Message processing.

3.7.5.1.2 On receipt of the first downlink CPDLC/IC Data value, the CPDLC-ground-user shall verify the Integrity Check verification procedure using:

a) the ATN Message Checksum verification procedure specified in Chapter 6 of this manual if the downlink CPDLC/IC Data does not contain an algorithm identifier, or contains the algorithm identifier "atn-default-checksum",

b) the specified algorithm if the downlink CPDLC/IC Data contains any other algorithm identifier.

3.7.5.1.3 On receipt of a subsequent downlink CPDLC/IC Data value, the CPDLC-ground-user shall:

a) ignore the specified algorithm if the downlink CPDLC/IC Data contains an algorithm identifier,

b) verify the Integrity CheckIntegrity Check using the Integrity Check verification procedure used for the preceding downlink CPDLC/IC Data.

3.7.5.1.4 On receipt of a downlink CPDLC/IC Data value containing an unrecognised Integrity Check algorithm identifier, the CPDLC-ground-user shall:

a) discard the received downlink CPDLC/IC Data, and

b) invoke the CPDLC-user-abort request primitive with the *Reason* parameter set to CPDLCUserAbortReason value [unknown-integrity-check].

Note.— This includes the case when a CPDLC-start or a DSC-start indication is received with a downlink CPDLC/IC Data value that includes an unrecognised algorithm identifier.

3.7.5.1.5 When the Integrity Check verification procedure fails, the CPDLC-ground-user shall:

a) discard the received downlink CPDLC/IC Data, and

b) invoke the CPDLC-user-abort request primitive with the *Reason* parameter set to CPDLCUserAbortReason value [validation-failure].

Note.— This includes the case when a CPDLC-start or a DSC-start indication is received with a downlink CPDLC/IC Data value and the Integrity Check verification procedure fails.

3.7.5.1.6 If the Integrity Check verification procedure of a downlink CPDLC/IC Data value succeeds, then the embeddedMessage element of the downlink CPDLC/IC Data shall be decoded as an ATCDownlinkMessage using unaligned basic PER, and processed by the CPDLC-ground-user, as specified in 3.7.6 and 3.7.8

3.7.5.1.7 If the CPDLC Message received in a downlink CPDLC/IC Data value cannot be successfully decoded as an ATCDownlinkMessage, then the CPDLC-ground-user shall:

a) discard the received downlink CPDLC Message, and

b) invoke the CPDLC-user-abort request primitive with the *Reason* parameter set to CPDLCUserAbortReason value [unable-to-decode-message].

3.7.5.2 Uplink CPDLC/IC Data

3.7.5.2.1 On receipt of an uplink CPDLC/IC Data value, the Integrity Check verification procedure shall be performed before any other CPDLC Message processing.

3.7.5.2.2 On receipt of the first uplink CPDLC/IC Data, the CPDLC-air-user shall verify the Integrity Check verification procedure using:

a) the ATN Message Checksum verification procedure specified in Chapter 6 of this manual, if the uplink CPDLC/IC Data does not contain an algorithm identifier, or contains the algorithm identifier "atn-default-checksum",

b) the specified algorithm if the uplink CPDLC/IC Data contains any other algorithm identifier.

3.7.5.2.3 On receipt of a subsequent uplink CPDLC/IC Data value, the CPDLC-air-user shall :

a) ignore the specified algorithm if the uplink CPDLC/IC Data contains an algorithm identifier,

b) verify the Integrity Check using the Integrity Check verification procedure used for the preceding uplink CPDLC/IC Data value.

3.7.5.2.4 On receipt of an uplink CPDLC/IC Data value containing an unrecognised Integrity Check algorithm identifier, the CPDLC-air-user shall:

a) discard the received uplink CPDLC/IC Data, and

b) invoke the CPDLC-user-abort request primitive with the *Reason* parameter set to CPDLCUserAbortReason value [unknown-integrity-check].

Note.— This includes the case when a CPDLC-start indication is received with an uplink CPDLC/IC Data value that includes an unrecognised algorithm identifier.

3.7.5.2.5 When the Integrity Check verification procedure fails, the CPDLC-air-user shall:

a) discard the received uplink CPDLC/IC Data, and

b) invoke the CPDLC-user-abort request primitive with the *Reason* parameter set to CPDLCUserAbortReason value [validation-failure].

Note.— This includes the case when a CPDLC-start Indication is received with an uplink CPDLC/IC Data value and the Integrity Check verification procedure fails.

3.7.5.2.6 If the Integrity Check verification procedure of an uplink CPDLC/IC Data value succeeds, then the embeddedMessage element of the uplink CPDLC/IC Data shall be decoded as an ATCUplinkMessage using unaligned basic PER, and processed by the CPDLC-air-user as specified in 3.7.6 and 3.7.7.

3.7.5.2.7 If the CPDLC Message received in an uplink CPDLC/IC Data value cannot be successfully decoded as an ATCUplinkMessage, then the CPDLC-air-user shall:

a) discard the received uplink CPDLC Message, and

b) invoke the CPDLC-user-abort request primitive with the *Reason* parameter set to CPDLCUserAbortReason value [unable-to-decode-message].

3.7.6 CPDLC Message Receipt Requirements

Note.— The CPDLC Message specified in this section is the uplink or downlink message exchanged between peer CPDLC-users. It is exchanged in PER-encoded form as the embeddedMessage element of a CPDLC/IC Data value. Before the CPDLC Message is processed as specified here, it will have successfully passed the Integrity Check verification procedure.

3.7.6.1 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.).

3.7.6.1.1 When a message contains a single message element, the message attributes shall be the message element attributes.

3.7.6.1.2 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.

3.7.6.2 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.

3.7.6.2.1 Each message element shall have associated Urgency attributes with precedence as defined in Table 3.7-1.

Table 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

3.7.6.2.2 When a CPDLC-user queues received messages, messages with the highest Urgency type shall be placed at the beginning of the queue.

3.7.6.2.3 When a CPDLC-user queues received messages, messages with the same Urgency type shall be queued in order of receipt.

3.7.6.3 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.

3.7.6.3.1 Each message element shall have associated Alert attributes with precedence as defined in Table 3.7-2.

Table 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

3.7.6.3.2 Upon receipt of a CPDLC Message, the CPDLC-user shall provide one of three distinct alerts as determined by the received message alert attribute.

3.7.6.4 Response Attribute

Note.— The response (RESP) attribute mandates CPDLC-user response requirements for a given message element. Response CPDLC/IC Data attribute only apply to air/ground messages.

3.7.6.4.1 Each uplink message element shall have associated Response attributes with precedence as defined in Table 3.7-3.

Table 3.7-3. Response Attribute (Up-Link)

Type	***Response Required***	***Valid Responses Description***	***Precedence***
W/U	*Yes*	*Response required: WILCO, UNABLE, STANDBY, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY, ERROR, LOGICAL ACKNOWLEDGEMENT (only if required)*	*1*
A/N	*Yes*	*Response required: AFFIRM, NEGATIVE, STANDBY, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY, ERROR, LOGICAL ACKNOWLEDGEMENT (only if required)*	*2*
R	*Yes*	*Response required: ROGER, UNABLE, STANDBY, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY, ERROR, LOGICAL ACKNOWLEDGEMENT (only if required),*	*3*
Y	*Yes*	*Any CPDLC downlink message, LOGICAL ACKNOWLEDGEMENT (only if required),*	*4*
N	*No, unless logical ACKNOWLEDGE MENT required*	*LOGICAL ACKNOWLEDGEMENT (only if required), ERROR, NOT CURRENT DATA AUTHORITY or NOT AUTHORIZED NEXT DATA AUTHORITY*	*5*

3.7.6.4.2 Each downlink message element shall have associated Response attributes with precedence as defined in Table 3.7-4.

Table 3.7-4. Response Attribute (Down-Link)

Type	***Response Required***	***Valid Responses Description***	***Precedence***
Y	*Yes*	*Any CPDLC uplink message,* *LOGICAL ACKNOWLEDGEMENT (only if required)*	*1*
N	*No, unless logical ACKNOWLEDGE MENT required*	*LOGICAL ACKNOWLEDGEMENT (only if required), ERROR, SERVICE UNAVAILABLE, or FLIGHT PLAN NOT HELD*	*2*

3.7.6.5 CPDLC/DSC Distinction

3.7.6.5.1 Upon receipt of a CPDLC Message the CPDLC-air-user shall provide a distinction between CPDLC Messages received from the Current Data Authority and those received from a Downstream Data Authority.

3.7.6.6 Air/Ground - Ground/Ground Distinction

3.7.6.6.1 [PDR 99010004/37]Upon receipt of a CPDLC Message the CPDLC-ground-user shall provide a distinction between CPDLC Messages received from an aircraft and those received from another ground system.

3.7.6.7 Logical ACKNOWLEDGEMENT Prohibited

3.7.6.7.1 Upon receipt of the CPDLC Message USE OF LOGICAL ACKNOWLEDGEMENT PROHIBITED the CPDLC-air-user shall be prohibited from requiring a logical ACKNOWLEDGEMENT for any message sent for the duration of the CPDLC or DSC dialogue.

3.7.6.7.2 If the CPDLC-ground-user receives a CPDLC Message requiring a logical ACKNOWLEDGEMENT where the use of logical ACKNOWLEDGEMENT 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 [logicalACKNOWLEDGEMENTNotAccepted] value in the value in the *CPDLC/IC Data* parameter and discard the content of the received message.

3.7.6.8 Message Reference Numbers

3.7.6.8.1 If a received CPDLC Message requires a response, the CPDLC-user shall provide a message reference number for each Response CPDLC/IC Data sent.

3.7.6.8.2 The message reference number shall be identical to the message identification number of the received message to which it refers.

3.7.6.9 Message Response Requirements

3.7.6.9.1 **Recommendation**.— *A message sequence initiated by data link should be closed by data link.*

3.7.6.9.2 **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.*

3.7.6.9.3 A CPDLC-user shall only be permitted to respond to a received message in its entirety.

3.7.6.9.4 Only one closure response shall be permitted for a given message.

3.7.6.9.5 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 CPDLC/IC Datas and then send a closure Response CPDLC/IC Data, or

b) send a closure Response CPDLC/IC Data.

3.7.6.9.6 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 CPDLC/IC Datas and then send a closure Response CPDLC/IC Data, or

b) send a closure Response CPDLC/IC Data.

3.7.6.9.7 For a given message, once the CPDLC-user has sent the closure Response CPDLC/IC Data, no other Response CPDLC/IC Datas shall be sent referring to the given message.

3.7.6.9.8 When a message is received by the CPDLC-user requiring a logical ACKNOWLEDGEMENT response,

a) when the CPDLC-user is a CPDLC-air-user, the CPDLC-air-user shall respond with either a CPDLC Message containing a LOGICAL ACKNOWLEDGEMENT message element, or with a message containing an ERROR, NOT CURRENT DATA AUTHORITY, or NOT AUTHORIZED NEXT DATA AUTHORITY message element as appropriate, or

b) when the CPDLC-user is a CPDLC-ground-user, the CPDLC-ground-user shall respond with a CPDLC Message containing a LOGICAL ACKNOWLEDGEMENT message element, or with a message containing an ERROR, SERVICE UNAVAILABLE, or FLIGHT PLAN NOT HELD message element as appropriate.

3.7.6.9.9 A logical ACKNOWLEDGEMENT Response CPDLC/IC Data, if required, shall be sent prior to sending any other related Response CPDLC/IC Data(s), except:

a) when the response is an uplink message, a Response CPDLC/IC Data containing an ERROR, SERVICE UNAVAILABLE, or FLIGHT PLAN NOT HELD message element as appropriate, or

b) when the response is a downlink message, a Response CPDLC/IC Data containing an ERROR, NOT CURRENT DATA AUTHORITY, or NOT AUTHORIZED NEXT DATA AUTHORITY message element as appropriate.

Note.— When case a) or b) occurs, the LOGICAL ACKNOWLEDGEMENT message element is not sent.

3.7.6.9.10 When CPDLC-air-user receives a message with a W/U RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGEMENT (if required), STANDBY, WILCO, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

3.7.6.9.11 When the CPDLC-air-user receives a message with a W/U RESP attribute, the closure Response CPDLC/IC Data shall contain at least a WILCO, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

3.7.6.9.12 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 ACKNOWLEDGEMENT (if required), STANDBY, AFFIRM, NEGATIVE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

3.7.6.9.13 When the CPDLC-air-user receives a message with an A/N RESP attribute, the closure Response CPDLC/IC Data shall contain at least a AFFIRM, NEGATIVE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

3.7.6.9.14 When the CPDLC-air-user receives a message with a R RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGEMENT (if required), STANDBY, ROGER, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

3.7.6.9.15 When the CPDLC-air-user receives a message with a R RESP attribute, the closure Response CPDLC/IC Data shall contain at least a ROGER, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

3.7.6.9.16 When the CPDLC-air-user receives a message with a Y RESP attribute, a LOGICAL ACKNOWLEDGEMENT only when requested, and all other CPDLC Messages shall be permitted as a Response CPDLC/IC Data.

3.7.6.9.17 When the CPDLC-air-user receives a message with a Y RESP attribute, the first Response CPDLC/IC Data sent that does not contain a STANDBY or LOGICAL ACKNOWLEDGEMENT shall constitute the closure Response CPDLC/IC Data.

3.7.6.9.18 When the CPDLC-ground-user receives an air/ground message with a Y RESP attribute, a LOGICAL ACKNOWLEDGEMENT, only when requested and all other CPDLC Messages shall be permitted as a Response CPDLC/IC Data.

3.7.6.9.19 When the CPDLC-ground-user receives an air/ground message with a Y RESP attribute, the first Response CPDLC/IC Data sent that does not contain a STANDBY, REQUEST DEFERRED, or LOGICAL ACKNOWLEDGEMENT message element shall constitute the closure Response CPDLC/IC Data.

3.7.6.9.20 When the CPDLC-air-user receives a message with a N RESP attribute, but requiring a logical ACKNOWLEDGEMENT, the only permitted response shall be a message that contains a LOGICAL ACKNOWLEDGEMENT, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element. This Response CPDLC/IC Data is the closure message.

3.7.6.9.21 When the CPDLC-ground-user receives an air/ground message with a N RESP attribute, but requiring a logical ACKNOWLEDGEMENT the only permitted response shall be a message that contains a LOGICAL ACKNOWLEDGEMENT, SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR message element. This Response CPDLC/IC Data is the closure message.

3.7.6.9.22 When the CPDLC-ground-user receives a ground/ground message the ground-user shall be prohibited from generating a ground/ground Response CPDLC/IC Data.

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.

3.7.6.10 Invalid Message Elements

3.7.6.10.1 The CPDLC-ground-user shall be prohibited from sending any CPDLC Message containing the uplink message elements 33, 40, 41, or 178.

3.7.6.11 Error Conditions

3.7.6.11.1 Duplicate Message Identification Numbers

3.7.6.11.1.1 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.

3.7.6.11.2 Invalid Reference Number

3.7.6.11.2.1 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 in the value in the CPDLC/IC Data parameter, and

b) disregard the received message.

3.7.6.11.3 No Available Message Identification Numbers

3.7.6.11.3.1 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.

3.7.6.11.4 Insufficient Resources

3.7.6.11.4.1 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 in the *CPDLC/IC Data* parameter, and

b) disregard the received message.

3.7.6.11.5 Invalid Message Element Combination

3.7.6.11.5.1 If a message is received containing:

a) a LOGICAL ACKNOWLEDGEMENT 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 with the [invalidMessageElementCombination] value in the *CPDLC/IC Data* parameter, and disregard the received message.

3.7.6.11.6 Invalid Message Elements

3.7.6.11.6.1 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 in the *CPDLC/IC Data* parameter, and disregard the received message.

3.7.6.11.7 System Management Responses

3.7.6.11.7.1 If the CPDLC-air-user sends a message containing the NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element instead of the expected Response CPDLC/IC Data, the NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message shall contain the received message identification number as the message reference number.

3.7.6.11.7.2 If the CPDLC-ground-user sends a message containing the SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR message element instead of the expected Response CPDLC/IC Data, the SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR message shall contain the received message identification number as the message reference number.

3.7.6.11.7.3 If the CPDLC-air-user sends a NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message in response to a CPDLC Message, the received CPDLC Message shall be disregarded.

3.7.6.11.7.4 If the CPDLC-ground-user sends a SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR message in response to a CPDLC Message, the received CPDLC Message shall be disregarded.

3.7.6.11.8 Invalid Message Response

3.7.6.11.8.1 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 ACKNOWLEDGEMENT, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY 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].

3.7.6.11.8.2 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 ACKNOWLEDGEMENT, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY 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].

3.7.6.11.8.3 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 ACKNOWLEDGEMENT, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY 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].

3.7.6.11.8.4 If the CPDLC-ground-user sends a message that has a N response attribute and requires a logical ACKNOWLEDGEMENT, and a response to this message is received by the CPDLC-ground-user that does not contain any of the following message elements: LOGICAL ACKNOWLEDGEMENT, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY 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].

3.7.6.11.8.5 If the CPDLC-air-user sends a message that has a N response attribute and requires a logical ACKNOWLEDGEMENT, and a response to this message is received by the CPDLC-air-user that does not contain any of the following message elements: LOGICAL ACKNOWLEDGEMENT, SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR [errorInformation] the CPDLC-air-user shall invoke CPDLC-user-abort request with a CPDLC Message containing the CPDLCUserAbortReason with the value [invalid-response].

3.7.6.11.9 Out of synchronisation

3.7.6.11.9.1 If a CPDLC-user receives a message containing a timestamp that is out of synchronisation with its own clock, the CPDLC-user shall invoke the CPDLC-user-abort request containing the CPDLCUserAbortReason with the value [time-out-of-synchronisation] as the Reason parameter.

Note.— The amount of time out of synchronisation that will trigger this event is a local implementation issue.

3.7.6.11.10 Preventing Error loops

3.7.6.11.10.1 If the CPDLC-air-user receives a message containing the ERROR message element, and the message is found to be in error, the CPDLC-air-user shall disregard the received message and be prohibited from responding with an ERROR message.

3.7.6.11.10.2 If the CPDLC-ground-user receives a message containing the ERROR message element, and the message is found to be in error, the CPDLC-ground-user shall disregard the received message and be prohibited from responding with an ERROR message.

Note 1.— This requirement refers to received ERROR messages that are recognized to be in error by the receiving system.

Note 2.— This requirement is to prevent getting into an infinite loop of sending errors.

3.7.7 CPDLC-air-user Requirements

3.7.7.1 The CPDLC-start Service

3.7.7.1.1 Invoking the CPDLC-start request

3.7.7.1.1.1 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.

3.7.7.1.1.2 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.

3.7.7.1.1.3 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.

3.7.7.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response

3.7.7.1.2.1 Upon receipt of a CPDLC-start service indication, the CPDLC-air-user shall invoke a CPDLC-start service response within 0.5 seconds.

3.7.7.1.2.2 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” and the *Response CPDLC/IC Data* parameter to an ICDownlinkMessage containing an embeddedMessage element LOGICAL ACKNOWLEDGEMENT if a logical ACKNOWLEDGEMENT was requested and the received CPDLC message is not in error, or the embeddedMessage element ERROR if the received CPDLC message is in error, 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 the Result parameter to the abstract value “rejected” and the Response CPDLC/IC Data parameter to an ICDownlinkMessage containing an

embeddedMessage with the message element NOT AUTHORIZED NEXT DATA AUTHORITY.

3.7.7.1.2.3 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.

3.7.7.1.2.4 If the CPDLC-air-user sets the CPDLC-start response Result parameter to the abstract value "rejected", any CPDLC Message contained in the CPDLC-start indication CPDLC/IC Data parameter shall be disregarded.

3.7.7.1.2.5 If the CPDLC-air-user sets the CPDLC-start 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/IC Data parameter shall be processed.

3.7.7.1.2.6 If the CPDLC-air-user sets the CPDLC-start 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/IC Data parameter shall be disregarded.

3.7.7.1.2.7 If the CPDLC-air-user sets the CPDLC-start 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 in the CPDLC-start indication Calling Peer Identifier parameter is the Next Data Authority associate the CPDLC-ASE invocation with the Next Data Authority.

3.7.7.1.2.8 If a 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.

3.7.7.1.3 Receipt of a CPDLC-start confirmation

3.7.7.1.3.1 If a CPDLC-start confirmation has been received with a Result parameter containing the abstract value "accepted" and the Response parameter contains no CPDLC message or a CPDLC message with either a LOGICAL ACKNOWLEDGEMENT message element or an ERROR message element, the CPDLC-air-user shall:

a) Establish an association between a CPDLC-ASE invocation and the ground system facility designation contained in the 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 in the CPDLC-start request Called Peer Identifier parameter is the Next Data Authority associate the CPDLC-ASE invocation with the Next Data Authority.

3.7.7.1.3.2 If a CPDLC-start confirmation has been received with a Result parameter containing the abstract value "accepted", and the Response parameter contains a CPDLC message with a message element other than a LOGICAL ACKNOWLEDGEMENT message element or an ERROR message element, the CPDLC-air-user shall invoke the CPDLC-user-abort request primitive with the Reason parameter set to CPDLCUserAbortReason value [invalid-CPDLC-message].

3.7.7.2 The DSC-start Service

3.7.7.2.1 Invoking the DSC-start request

3.7.7.2.1.1 Only a CPDLC-air-user shall be permitted to invoke the DSC-start service request primitive.

3.7.7.2.1.2 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.

3.7.7.2.1.3 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.

3.7.7.2.2 Receipt of a DSC-start confirmation

3.7.7.2.2.1 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 the DSC-start request Facility Designation parameter,

b) Associate the CPDLC-ASE invocation with a Downstream Data Authority.

3.7.7.3 The CPDLC-message Service

3.7.7.3.1 Invoking the CPDLC-message request

3.7.7.3.1.1 The CPDLC-air-user shall be prohibited from invoking the CPDLC-message service request

primitive with a CPDLC/IC Data parameter not containing a CPDLC Message.

3.7.7.3.2 Receipt of a CPDLC-message Indication

3.7.7.3.2.1 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 embeddedMessage contained in the CPDLC/IC Data parameter.

3.7.7.3.2.2 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.

3.7.7.3.2.3 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.

3.7.7.3.2.4 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 CPDLC Message shall be disregarded.

3.7.7.3.2.5 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/IC Data parameter containing a CPDLC Message with the message element NOT CURRENT DATA AUTHORITY, and

b) Disregard the received CPDLC Message contained in the CPDLC-message indication CPDLC/IC Data parameter.

3.7.7.3.2.6 If a CPDLC-message indication is received containing a CPDLC/IC Data parameter with no embeddedMessage element, the CPDLC-air-user shall invoke CPDLC-user-abort request with the Reason parameter set to CPDLCUserAbortReason value [invalid-pdu].

3.7.7.4 The CPDLC-end Service

3.7.7.4.1 The CPDLC-end Service Request

3.7.7.4.1.1 The CPDLC-air-user shall be prohibited from invoking the CPDLC-end request.

3.7.7.4.2 Receipt of a CPDLC-end Indication and Invoking a CPDLC-end Response

3.7.7.4.2.1 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/IC Data parameter containing a CPDLC Message with the message element NOT CURRENT DATA AUTHORITY, and

2) the Result parameter set to the abstract value "rejected", and

b) Disregard any CPDLC Message provided in the CPDLC-end indication CPDLC/IC Data parameter.

Note 3.— 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 4.— 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 5.— 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 6.— 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 parameter set to the abstract value "accepted" on the ground side.

Note 7.— Local procedures may dictate when a "rejected" response Result parameter is permitted by the CPDLC-air-user in the cases when section 3.7 gives the CPDLC-air-user a choice between "accepted" or "rejected".

Note 8.— If a CPDLC-end service response is invoked with a "accepted" Result this will result in ending the dialogue regardless of any CPDLC Messages contained in the CPDLC/IC Data parameter.

3.7.7.4.2.2 Uplink Message Contains Error

3.7.7.4.2.2.1 If a CPDLC-end indication is received from the Current Data Authority and there is a CPDLC Message in the CPDLC/IC Data parameter that either has the response attribute N and requires a logical ACKNOWLEDGEMENT 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/IC Data parameter containing a CPDLC Message with the message element ERROR [errorInformation],

b) the Result parameter set to the abstract value "rejected".

3.7.7.4.2.3 No CPDLC Message in the CPDLC-end Indication

3.7.7.4.2.3.1 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 CPDLC Message in the CPDLC/IC Data parameter, then the CPDLC-air-user shall invoke CPDLC-end response with the Result parameter set to the abstract value "accepted".

3.7.7.4.2.3.2 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 CPDLC message in the CPDLC/IC Data parameter, then the CPDLC-air-user shall invoke CPDLC-end response with the Result parameter set to the abstract value "accepted" or "rejected".

3.7.7.4.2.4 Message in CPDLC-end Indication Does Not Require Response

3.7.7.4.2.4.1 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 CPDLC message in the CPDLC/IC Data parameter with the response attribute N and not requiring a logical ACKNOWLEDGEMENT, then the CPDLC-air-user shall invoke CPDLC-end response with the Result parameter set to the abstract value "accepted".

3.7.7.4.2.4.2 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 CPDLC Message in the CPDLC/IC Data parameter with the response attribute N and not requiring a logical ACKNOWLEDGEMENT, then the CPDLC-air-user shall invoke CPDLC-end response with the Result parameter set to the abstract value "accepted" or "rejected".

3.7.7.4.2.4.3 CPDLC Message in CPDLC-end Indication Requires Only Logical ACKNOWLEDGEMENT.

3.7.7.4.2.4.4 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 CPDLC message in the CPDLC/IC Data parameter with the response attribute N and requiring a logical ACKNOWLEDGEMENT, and no error is detected in the CPDLC message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the CPDLC/IC Data parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGEMENT, and

b) the Result parameter set to the abstract value "accepted".

3.7.7.4.2.4.5 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 CPDLC Message in the CPDLC/IC Data parameter with the response attribute N and requiring a logical ACKNOWLEDGEMENT, and no error is detected in the CPDLC message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the CPDLC/IC Data parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGEMENT, and

b) the Result parameter set to the abstract value "accepted" or "rejected".

3.7.7.4.2.2.4.6 CPDLC Message in CPDLC-end Indication With W/U, A/N, or R Attribute, Positive Response

Note.— In this case, positive response to a CPDLC Message in the CPDLC-end Indication also indicates acceptance of the end of the dialogue.

3.7.7.4.2.2.4.7 If a CPDLC-end indication is received from the Current Data Authority and there is a CPDLC message in the *CPDLC/IC Data* 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 CPDLC message, then the CPDLC-air-user shall:

a) if a logical ACKNOWLEDGEMENT is required, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGEMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the CPDLC/IC Data 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".

3.7.7.4.2.2.4.8 CPDLC 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.

3.7.7.4.2.2.4.8 If a CPDLC-end indication is received from the Current Data Authority and there is a CPDLC Message in the CPDLC/IC Data 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 CPDLC Message, then the CPDLC-air-user shall:

a) if a logical ACKNOWLEDGEMENT is required, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGEMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the CPDLC/IC Data 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”.

3.7.7.4.2.3 CPDLC Message in CPDLC-end Indication With Y Response Attribute

3.7.7.4.3.1.1 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 CPDLC Message in the CPDLC/IC Data parameter with the response attribute Y and no error is detected in the CPDLC Message, the CPDLC-air-user shall:

a) if a logical ACKNOWLEDGEMENT is required, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGEMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the CPDLC/IC Data parameter containing a CPDLC Message with a Y attribute closure CPDLC Message, and

2) the Result parameter set to the abstract value "accepted".

3.7.7.4.3.1.2 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 CPDLC Message in the CPDLC/IC Data parameter with the response attribute Y and no error is detected in the CPDLC Message, the CPDLC-air-user shall:

a) if a logical ACKNOWLEDGEMENT is required, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGEMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the CPDLC/IC Data parameter containing CPDLC Message with a Y attribute closure CPDLC Message, and

2) the Result parameter set to the abstract value "accepted" or "rejected".

3.7.7.4.3.1.3 Upon invoking a 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.

3.7.7.4.3.1.4 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.

3.7.7.5 The DSC-end Service

3.7.7.5.1 The DSC-end Request

3.7.7.5.1.1 Only the CPDLC-air-user shall be permitted to invoke the DSC-end request.

3.7.7.5.1.2 If a CPDLC-air-user has invoked a DSC-end service request primitive, the CPDLC-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.

3.7.7.6 The CPDLC-user-abort Service

3.7.7.6.1 Issuing a CPDLC-user-abort Request [commanded-termination]

3.7.7.6.1.1 The CPDLC-air-user shall have the capability to invoke CPDLC-user-abort request with the Reason parameter set to CPDLCUserAbortReason value [commanded-termination].

3.7.7.6.2 Invoking a CPDLC-user-abort Request

3.7.7.6.2.1 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.

3.7.7.6.2.2 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.

3.7.7.6.3 Receipt of a CPDLC-abort Indication

3.7.7.6.3.1 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.

3.7.7.6.3.2 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.

3.7.8 CPDLC-ground-user Requirements

3.7.8.1 The CPDLC-start Service

3.7.8.1.1 Invoking the CPDLC-start request

3.7.8.1.1.2 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.

3.7.8.1.1.3 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 initiated by the CPDLC-start service request, except the CPDLC-user-abort request until after it has received a CPDLC-start confirmation.

3.7.8.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response

3.7.8.1.2.1 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.

3.7.8.1.2.2 The CPDLC-ground-user shall be prohibited from invoking the CPDLC-start response unless and until it has received a CPDLC-start indication.

3.7.8.1.2.3 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value "rejected" then the *Response CPDLC/IC Data* parameter shall be an ICUplinkMessage containing a CPDLC Message with either the SERVICE UNAVAILABLE or FLIGHT PLAN NOT HELD message element as appropriate.

3.7.8.1.2.4 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/IC Data* parameter.

3.7.8.1.2.5 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value "accepted" any CPDLC Message contained in the CPDLC-start indication *CPDLC/IC Data* parameter shall be processed.

3.7.8.1.2.6 When the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value "accepted", it shall set the *Response CPDLC/IC Data* parameter to an ICUplinkMessage containing:

a) A CPDLC Message with the message element LOGICAL ACKNOWLEDGEMENT if a logical acknowledgement was requested and the CPDLC message received is not in error,

b) A CPDLC Message with the message element ERROR if the CPDLC message received is in error, or

c) No CPDLC Message in all other cases.

3.7.8.1.2.7 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value “accepted” the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and the ICAO 24 bit aircraft address contained in CPDLC-start indication Calling Peer Identifier parameter.

3.7.8.1.2.8 If a 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.

3.7.8.1.2.9 Upon receipt of a CPDLC-start service indication, the CPDLC-ground-user shall invoke a CPDLC-start service response within 0.5 seconds.

3.7.8.1.3 Receipt of a CPDLC-start confirmation

3.7.8.1.3.1 If a CPDLC-start confirmation has been received with a Result parameter containing the abstract value “accepted”, and the Response parameter contains no CPDLC message or a CPDLC message with either a LOGICAL ACKNOWLEDGEMENT message element, an ERROR message element, or a NOT CURRENT DATA AUTHORITY message element, the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and the ICAO 24 bit aircraft address contained in CPDLC-start request Called Peer Identifier parameter.

3.7.8.1.3.2 If a CPDLC-start confirmation has been received with a Result parameter containing the abstract value "accepted", and the Response parameter contains a CPDLC message a message element other than a LOGICAL ACKNOWLEDGEMENT message element, an ERROR message element, or a NOT CURRENT DATA AUTHORITY message element, the CPDLC-ground-user shall invoke the CPDLC-user-abort request primitive with the Reason parameter set to CPDLCUserAbortReason value [invalid-CPDLC-message].

3.7.8.2 The DSC-start Service

3.7.8.2.1 Receipt of a DSC-start Indication and Invoking DSC-start Response

3.7.8.2.1.2 The CPDLC-ground-user shall be prohibited from invoking the DSC-start response unless and until it has received a DSC-start indication.

3.7.8.2.1.3 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.

3.7.8.2.1.4 If the CPDLC-ground-user sets the DSC-start response Result parameter to the abstract value "rejected" then the Response CPDLC/IC Data parameter shall be an ICUplinkMessage containing a CPDLC Message with either the SERVICE UNAVAILABLE or FLIGHT PLAN NOT HELD message element as appropriate.

3.7.8.2.1.5 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/IC Data parameter.

3.7.8.2.1.6 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/IC Data parameter shall be processed.

3.7.8.2.1.7 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 the ICAO 24 bit aircraft address contained in the DSC-start indication Aircraft Address parameter.

3.7.8.2.1.8 If a 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.

3.7.8.2.2 Receipt of a DSC-start Indication and Invoking a DSC-start Response

3.7.8.2.2.1 Upon receipt of a DSC-start indication, the CPDLC-ground-user shall invoke a DSC-start response within 0.5 seconds.

3.7.8.2.2.2 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 Response CPDLC/IC Data parameter to an ICUplinkMessage containing a CPDLC Message with the message element SERVICE UNAVAILABLE or FLIGHT PLAN NOT HELD as appropriate.

3.7.8.3 The CPDLC-message Service

3.7.8.3.1 Invoking the CPDLC-message request

3.7.8.3.1.1 The CPDLC-ground-user shall be prohibited from invoking the CPDLC-message service

request primitive with a CPDLC/IC Data parameter not containing a CPDLC Message.

3.7.8.3.2 Receipt of a CPDLC-message indication

3.7.8.3.2.1 If a CPDLC-message indication is received containing a CPDLC/IC Data parameter with no embeddedMessage element, the CPDLC-ground-user shall invoke CPDLC-user-abort request with the Reason parameter set to CPDLCUserAbortReason value [invalid-pdu].

3.7.8.4 The CPDLC-end Service

3.7.8.4.1 The CPDLC-end Request

3.7.8.4.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-end request.

3.7.8.4.1.2 If a CPDLC-ground-user has invoked a CPDLC-end service request primitive, the CPDLC-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.

3.7.8.5 The DSC-end Service

3.7.8.5.1 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 CPDLC 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 section 3.7 gives 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 CPDLC Messages contained in the CPDLC/IC Data e parameter.

3.7.8.5.1.1 Downlink Message Contains Error

3.7.8.5.1.2 If a DSC-end indication is received and there is a CPDLC Message in the *CPDLC/IC Data* parameter that either has the response attribute N and requires a logical ACKNOWLEDGEMENT or has Y response attribute, and an error is detected in the CPDLC Message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the CPDLC/IC Data parameter containing CPDLC Message with the message element ERROR [errorInformation],

b) the Result parameter set to the abstract value "rejected".

3.7.8.5.1.3 No CPDLC Message in the DSC-end Indication

3.7.8.5.1.3.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is no CPDLC Message in the CPDLC/IC Data parameter, then the CPDLC-ground-user shall invoke DSC-end response with the Result parameter set to the abstract value "accepted".

3.7.8.5.1.3.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is no CPDLC Message in the CPDLC/IC Data parameter, then the CPDLC-ground-user shall invoke DSC-end response with the Result parameter set to the abstract value "accepted" or "rejected".

3.7.8.5.1.4 CPDLC Message in DSC-end Indication Does Not Require Response

3.7.8.5.1.4.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a CPDLC Message in the CPDLC/IC Data parameter with the response attribute N and not requiring a logical ACKNOWLEDGEMENT, then the CPDLC-ground-user shall invoke DSC-end response with the Result parameter set to the abstract value "accepted".

3.7.8.5.1.4.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a CPDLC Message in the CPDLC/IC Data parameter with the response attribute N and not requiring a logical ACKNOWLEDGEMENT, then the CPDLC-ground-user shall invoke DSC-end response with the Result parameter set to the abstract value "accepted" or "rejected".

3.7.8.5.1.5 CPDLC Message in DSC-end Indication Requires Only Logical ACKNOWLEDGEMENT

3.7.8.5.1.5.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a CPDLC Message in the CPDLC/IC Data parameter with the response attribute N and requiring a logical ACKNOWLEDGEMENT, and no error is detected in the CPDLC Message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the CPDLC/IC Data parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGEMENT, and

b) the Result parameter set to the abstract value "accepted".

If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a CPDLC Message in the CPDLC/IC Data parameter with the response attribute N and requiring a logical ACKNOWLEDGEMENT, and no error is detected in the CPDLC Message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the CPDLC/IC Data parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGEMENT, and

b) the Result parameter set to the abstract value "accepted" or "rejected".

3.7.8.5.1.5.2 CPDLC Message in DSC-end Indication With Y Response Attribute

If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a CPDLC Message in the CPDLC/IC Data parameter with the response attribute Y and no error is detected in the CPDLC Message the CPDLC-ground-user shall:

a) if a logical ACKNOWLEDGEMENT is required, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGEMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the CPDLC/IC Data 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/IC Data parameter containing a CPDLC Message with at least the message element REQUEST DEFERRED, and

d) invoke DSC-end response with:

1) the CPDLC/IC Data parameter containing a CPDLC Message with a Y attribute closure CPDLC Message, and

2) the Result parameter set to the abstract value "accepted".

If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a CPDLC Message in the CPDLC/IC Data parameter with the response attribute Y and no error is detected in the message the CPDLC-ground-user shall:

a) if a logical ACKNOWLEDGEMENT is required, invoke CPDLC-message request with the CPDLC/IC Data parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGEMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the CPDLC/IC Data 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/IC Data parameter containing a CPDLC Message with at least the message element REQUEST DEFERRED, and

d) invoke DSC-end response with:

1) the CPDLC/IC Data parameter containing a CPDLC Message with a Y attribute closure CPDLC Message, and

2) the Result parameter set to the abstract value "accepted" or "rejected".

3.7.8.6 The CPDLC-forward Service

3.7.8.6.1 Invoking the CPDLC-forward Request

3.7.8.6.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-forward Request.

3.7.8.6.1.2 The CPDLC-ground-user shall include in the CPDLC-forward request CPDLC Message parameter a MessageForward APDU-element containing an ATCDownlinkMessageData or an ATCUplinkMessageData encoded using the unaligned basic PER.

3.7.8.7 The CPDLC-user-abort Service

3.7.8.7.1 Issuing a CPDLC-user-abort Request

3.7.8.7.1.1 The CPDLC-ground-user shall have the capability to invoke CPDLC-user-abort request with the Reason parameter set to CPDLCUserAbortReason value [commanded-termination].

3.7.9 Message Intent

3.7.9.1 Purpose

Note 1.— This section specifies the message set for CPDLC Message attributes, message presentation guidance, and data structure presentation guidance. The actual information exchanged between an aircraft and ground peer or between two ground peer CPDLC applications is defined in 3.4; however, 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 matter. The message presentation recommendations contained in ICAO Doc 4444 are one possible means of presenting the information.

3.7.9.2 Uplink and downlink message elements in this manual comply with the provisions of the ICAO *Procedures for Air Navigation Services — Air Traffic Management* (PANS-ATM, Doc 4444), 14th edition (2001), Appendix 5. Later amendments to this material are not included in this manual.

3.7.10 Parameter Value Unit, Range, and Resolution

A CPDLC-user shall interpret CPDLC Message element variables unit, range, and resolution as defined in 3.4.

— — — — — — — —

3.8 SUBSETTING RULES

3.8.1 General

Note.— This chapter specifies conformance requirements which all implementations of the CPDLC protocol obey.

3.8.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service claiming conformance to this chapter 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 3.8-1. Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 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 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 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
V.	CPDLC/ground + DSC-FU FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus:

	List of Predicates	**Functionality Description**
		· 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)

— — — — — — — —

Chapter 6

6. ATN MESSAGE INTEGRITY CHECK ALGORITHM

6.1 Use of Integrity Check Function

Note.— The integrity check algorithm specified in this section may optionally be invoked by the ATN Application User specification to provide a proof of message integrity. If other information (e.g. sender and/or receiver identity) is also bound to the message when the integrity sequence is computed and verified then it can also provide additional proofs (e.g. a proof of delivery to an intended recipient).

6.1.1 By default, an application Integrity Check shall be computed using the Default ATN Message Checksum algorithm specified in this chapter.

Note.— Other Integrity Check algorithms may be used and their use identified by including, in the application message, a Relative OID that identifies the algorithm alongside the computed Integrity Check. However, the use of an alternative Integrity Check algorithm must be agreed in advance by both air and ground users using a procedure outside of the scope of this specification.

6.2 Default ATN Message Checksum Algorithm

Note 1.— The Default ATN Message Checksum algorithm generates a 32-bit (4 octet) checksum.

Note 2.— The Default ATN Message Checksum algorithm is a 32-bit version of the Fletcher algorithm. This has been chosen in order to satisfy requirements for high data integrity with good performance characteristics.

Note 3.— The object identifier "atn-default-checksum" as defined below identifies the Default ATN Message Checksum algorithm :

atn-default-checksum ::= OBJECT IDENTIFIER {iso(1) identified organization(3) icao(27) atn-algorithms(9) atc-chk32(0)}

6.2.1 Symbols

Note 1.— The symbols used in this specification are as follows:

C0, C1, C2, C3 are variables used by the algorithm.

i is the number (i.e. position) of an octet within the Message to be protected

L is the length of the Message to be protected, in octets.

X_j is the value of the j'th octet of the checksum BIT STRING, where X_0 is the first octet in the BIT STRING, X_1 is the second, X_2 is the third and X_3 is the fourth octet.

Note 2.— The data to be protected is a bit string provided by the user of this algorithm .

6.2.2 Arithmetic Conventions

6.2.2.1 Addition shall be performed in one of the two following modes:

a) modulo 255, or

b) ones complement arithmetic.

6.2.2.2 In the ones complement mode, if any of the variables has the value minus zero (i.e. 0xFFFF), it shall be regarded as though it were plus zero (i.e. 0).

6.2.3 Algorithm for Checksum Generation

6.2.3.1 If the data to be protected is not an integral number of octets, it shall be right padded to the next octet boundary by appending bits with value 'zero' (0).

6.2.3.2 The ATN Message checksum shall be generated by the following algorithm.

a) Initialise C0, C1, C2 and C3 to zero

b) Process each octet in the Message sequentially from i = 1 to L by

1) **adding the value of the octet to C0; then**

2) **adding the value of C0 to C1, C1 to C2, and C2 to C3**

c) Set the octets of the checksum as follows:

1) **X0 = - (C0 + C1 + C2 + C3)**

2) **X1 = C1 + 2*C2 + 3*C3**

3) **X2 = - (C2 + 3*C3)**

4) **X_3 = C3**

6.2.4 Algorithm for Checksum Verification

6.2.4.1 When the integrity of a message is to be verified using this algorithm, the verification function shall:

a) Append to the Message the octets of the checksum field in the same order in which they appear in the checksum BIT STRING.

b) Initialise C0, C1, C2 and C3 to zero

c) Process each octet in the Message, including the appended checksum octets, sequentially from i = 1 to L by

   1) adding the value of the octet to C0; then

   2) adding the value of C0 to C1, C1 to C2, and C2 to C3.

d) Discard the appended checksum octets.

e) If, when all the octets have been processed, all of the variables C0, C1, C2 and C3 have the value zero, then a "success" indication is given to the algorithm user. Otherwise, a "verification failure" indication is given to the algorithm user.

Manual of Technical Provisions for the Aeronautical Telecommunication Network (ATN)

(Doc 9705-AN/956)

THIRD EDITION – 2002

SUB-VOLUME II – AIR GROUND APPLICATIONS

1. Sub-volume II of Doc 9705 has been replaced in its entirety by the *Manual on detailed technical specifications for aeronautical telecommunication network (ATN) using ISO/OSI standards and protocols* (Doc 9880-AN/466) – Part I.

2. Amendments to Doc 9705 were necessary to introduce an integrity check to the material for controller-pilot data link communications (CPDLC), flight information service (FIS) and automatic dependent surveillance-broadcast (ADS-B). Material for FIS and ADS-B will be introduced in Doc 9880 is being developed.

3. A table, mapping the provisions of Doc 9705 to those of Doc 9880 is included.

4. Doc 9880 has been approved by the Secretary-General. Pending its final editing and publication by ICAO it is available as draft Doc 9880 for use by the members of the Aeronautical Telecommunications Panel.

Doc 9880 (1st edition)	**Paragraph numbering as in Doc 9705 – third edition (2002)**
Chapter 2	*2.1 CONTEXT MANAGEMENT APPLICATION*
2.1	*2.1.1 INTRODUCTION*
1.4.1	*2.1.1.1*
2.1.1	*Note 1 Structure*
2.1.2	*Note 2 Functional description*
2.1.2.1	*Note 2 a) Logon functional description*
2.1.2.1.1	*Note 2 a) 1)*
2.1.2.1.2	*Note 2 a) 2)*
2.1.2.1.3	*Note 2 a) 3)*
2.1.2.1.4	*Note 2 a) 4)*
2.1.2.1.5	*Note 2 a) 5)*
2.1.2.1.6	*Note 2 a) 6)*
2.1.2.1.7	*Note 2 a) 7)*
2.1.2.2	*Note 2 b) Server facility query functional description*
2.1.2.2.1	*Note 2 b) 1)*
2.1.2.2.2	*Note 2 b) 2)*
2.1.2.2.3	*Note 2 b) 3)*
2.1.2.2.4	*Note 2 b) 4)*
2.1.2.2.5	*Note 2 b) 5)*
2.1.2.3	*Note 2 c) Server facility update functional description*
2.1.2.3.1	*Note 2 c) 1)*
2.1.2.3.2	*Note 2 c) 2)*
2.1.2.4	*Note 2 d) Update functional description*
2.1.2.4.1	*Note 2 d) 1)*
2.1.2.4.2	*Note 2 d) 2)*
2.1.2.4.3	*Note 2 d) 3)*
2.1.2.5	*Note 2 e) Contact functional description*
2.1.2.5.1	*Note 2 e) 1)*
2.1.2.5.2	*Note 2 e) 2)*
2.1.2.5.3	*Note 2 e) 3)*
2.1.2.6	*Note 2 f) Forwarding functional description*
2.1.2.6.1	*Note 2 f) 1)*
2.1.2.6.2	*Note 2 f) 2)*
2.1.2.6.3	*Note 2 f) 3)*
2.1.2.7	*Note 2 g) Registration functional description*
2.1.2.7.1	*Note 2 g) 1)*
2.1.2.7.2	*Note 2 g) 2)*
2.1.2.7.3	*Note 2 g) 3)*
2.2	*2.1.2 GENERAL REQUIREMENTS*
2.2.1	*2.1.2.1 CM ASE Version Number*
2.2.1.1	*2.1.2.1.1*
2.2.2	*2.1.2.2 Error Processing Requirements*
2.2.2.1	*2.1.2.2.1*
2.1.2.2.	*2.1.2.2.2*
2.3	*2.1.3 THE ABSTRACT SERVICE*
2.3.1	*2.1.3.1 Service Description*
2.3.1.1	*2.1.3.1.1*
2.3.2	*2.1.3.2 The CM-ASE Abstract Service*
2.3.2.1	*2.1.3.2.1*
2.3.3	*2.1.3.3 CM-logon Service*
2.3.3.1	*2.1.3.3.1*
2.3.3.2	*2.1.3.3.2*
2.3.3.3	*2.1.3.3.3 Facility designation*
2.3.3.3.1	*2.1.3.3.3.1*
2.3.3.4	*2.1.3.3.4 Aircraft Address*

2.3.3.4.1	*2.1.3.3.4.1*
2.3.3.5	*2.1.3.3.5 CM ASE Version Number*
2.3.3.5.1	*2.1.3.3.5.1*
2.3.3.5.2	*2.1.3.3.5.2*
2.3.3.5.3	*2.1.3.3.5.3*
2.3.3.6	*2.1.3.3.6 Logon Request*
2.3.3.6.1	*2.1.3.3.6.1*
2.3.3.7	*2.1.3.3.7 Logon Response*
2.3.3.7.1	*2.1.3.3.7.1*
2.3.3.7.2	*2.1.3.3.7.2*
2.3.3.8	*2.1.3.3.8 Class of Communication Service*
2.3.3.8.1	*2.1.3.3.8.1*
2.3.3.9	*2.1.3.3.9 Maintain Dialogue*
2.3.3.9.1	*2.1.3.3.9.1*
2.3.3.10	*2.1.3.3.10 Security Required*
2.3.3.10.1	*2.1.3.3.10.1*
2.3.3.10.2	*2.1.3.3.10.2*
2.3.3.10.3	*2.1.3.3.10.3*
2.3.3.11	*2.1.3.3.11 Emulated Version*
2.3.3.11.1	*2.1.3.3.11.1*
2.3.4	*2.1.3.4 CM-server-facility-query Service*
2.3.4.1	*2.1.3.4.1*
2.3.4.2	*2.1.3.4.2 Facility designation*
2.3.4.2.1	*2.1.3.4.2.1*
2.3.4.2.2	*2.1.3.4.2.2*
2.3.4.3	*2.1.3.4.3 Aircraft Address*
2.3.4.3.1	*2.1.3.4.3.1*
2.3.4.3.2	*2.1.3.4.3.2*
2.3.4.4	*2.1.3.4.4 Server Facility request*
2.3.4.4.1	*2.1.3.4.4.1*
2.3.4.5	*2.1.3.4.5 Server Facility Query Response*
2.3.4.5.1	*2.1.3.4.5.1*
2.3.4.6	*2.1.3.4.6 Class of Communication Service*
2.3.4.6.1	*2.1.3.4.6.1*
2.3.4.7	*2.1.3.4.7 Maintain Dialogue*
2.3.4.7.1	*2.1.3.4.7.1*
2.3.4.8	*2.1.3.4.8 Security Required*
2.3.4.8.1	*2.1.3.4.8.1*
2.3.4.8.2	*2.1.3.4.8.2*
2.3.4.8.3	*2.1.3.4.8.3*
2.3.4.8.4	*2.1.3.4.8.4*
2.3.4.8.5	*2.1.3.4.8.5*
2.3.5	*2.1.3.5 CM-update Service*
2.3.5.1	*2.1.3.5.1*
2.3.5.2	*2.1.3.5.2 Aircraft Address*
2.3.5.2.1	*2.1.3.5.2.1*
2.3.5.2.2	*2.1.3.5.2.2*
2.3.5.3	*2.1.3.5.3 Facility Designation*
2.3.5.3.1	*2.1.3.5.3.1*
2.3.5.3.2	*2.1.3.5.3.2*
2.3.5.4	*2.1.3.5.4 Update Information*
2.3.5.4.1	*2.1.3.5.4.1*
2.3.5.4.2	*2.1.3.5.4.2*
2.3.5.5	*2.1.3.5.5 Class of Communication Service*
2.3.5.5.1	*2.1.3.5.5.1*
2.3.5.6	*2.1.3.5.6 Security Required*
2.3.5.6.1	*2.1.3.5.6.1*
2.3.5.6.2	*2.1.3.5.6.2*

2.3.5.6.3	*2.1.3.5.6.3*
2.3.5.6.4	*2.1.3.5.6.4*
2.3.5.6.5	*2.1.3.5.6.5*
2.3.6	*2.1.3.6 CM-server-facility-update Service*
2.3.6.1	*2.1.3.6.1*
2.3.6.2	*2.1.3.6.2 Aircraft Address*
2.3.6.2.1	*2.1.3.6.2.1*
2.3.6.2.2	*2.1.3.6.2.2*
2.3.6.3	*2.1.3.6.3 Facility Designation*
2.3.6.3.1	*2.1.3.6.3.1*
2.3.6.3.2	*2.1.3.6.3.2*
2.3.6.4	*2.1.3.6.4 Server Facility Update Information*
2.3.6.4.1	*2.1.3.6.4.1*
2.3.6.5	*2.1.3.6.5 Class of Communication Service*
2.3.6.5.1	*2.1.3.6.5.1*
2.3.6.6	*2.1.3.6.6 Security Required*
2.3.6.6.1	*2.1.3.6.6.1*
2.3.6.6.2	*2.1.3.6.6.2*
2.3.6.6.3	*2.1.3.6.6.3*
2.3.6.6.4	*2.1.3.6.6.4*
2.3.7	*2.1.3.7 CM-contact Service*
2.3.7.1	*2.1.3.7.1*
2.3.7.2	*2.1.3.7.2 Aircraft Address*
2.3.7.2.1	*2.1.3.7.2.1*
2.3.7.2.2	*2.1.3.7.2.2*
2.3.7.3	*2.1.3.7.3 Facility Designation*
2.3.7.3.1	*2.1.3.7.3.1*
2.3.7.3.2	*2.1.3.7.3.2*
2.3.7.4	*2.1.3.7.4 Contact Request*
2.3.7.4.1	*2.1.3.7.4.1*
2.3.7.5	*2.1.3.7.5 Contact Response*
2.3.7.5.1	*2.1.3.7.5.1*
2.3.7.6	*2.1.3.7.6 Class of Communication Service*
2.3.7.6.1	*2.1.3.7.6.1*
2.3.7.7	*2.1.3.7.7 Security Required*
	2.1.3.7.7.1
2.3.7.7.1	*2.1.3.7.7.2*
2.3.7.7.2	*2.1.3.7.7.3*
2.3.7.7.3	*2.1.3.7.7.4*
2.3.8	*2.1.3.8 CM-end Service*
2.3.8.1	*2.1.3.8.1*
2.3.9	*2.1.3.9 CM-forward Service*
2.3.9.1	*2.1.3.9.1*
2.3.9.2	*2.1.3.9.2*
2.3.9.3	*2.1.3.9.3 Called Facility Designation*
2.3.9.3.1	*2.1.3.9.3.1*
2.3.9.4	*2.1.3.9.4 Calling Facility Designation*
2.3.9.4.1	*2.1.3.9.4.1*
2.3.9.5	*2.1.3.9.5 CM ASE Version Number*
2.3.9.5.1	*2.1.3.9.5.1*
2.3.9.5.2	*2.1.3.9.5.2*
2.3.9.5.3	*2.1.3.9.5.3*
2.3.9.6	*2.1.3.9.6 Forward Request*
2.3.9.6.1	*2.1.3.9.6.1*
2.3.9.6.2	*2.1.3.9.6.2*
2.3.9.7	*2.1.3.9.7 Class of Communication Service*
2.3.9.7.1.	*2.1.3.9.7.1.*
2.3.9.8	*2.1.3.9.8 Result*

2.3.9.8.1	*2.1.3.9.8.1*
2.3.9.9	*2.1.3.9.9 Security Required*
2.3.9.9.1	*2.1.3.9.9.1*
2.3.9.9.2	*2.1.3.9.9.2*
2.3.9.10	*2.1.3.9.10 Emulated Version*
2.3.9.10.1	*2.1.3.9.10.1*
2.3.10	*2.1.3.10 CM-user-abort Service*
2.3.10.1	*2.1.3.10.1*
2.3.11	*2.1.3.11 CM-provider-abort Service*
2.3.11.1	*2.1.3.11.1*
2.3.11.2	*2.1.3.11.2 Reason*
2.3.11.2.1	*2.1.3.11.2.1*
2.4	*2.1.4 FORMAL DEFINITIONS OF MESSAGES*
2.4.1	*2.1.4.1 Encoding/decoding Rules*
2.4.1.1	*2.1.4.1.1*
2.4.1.2	*2.1.4.1.2*
2.4.2	*2.1.4.2 CM ASN.1 Abstract Syntax*
2.4.2.1	*2.1.4.2.1*
2.5	*2.1.5 PROTOCOL DEFINITION*
2.5.1	*2.1.5.1 Sequence Rules*
2.5.1.1	*2.1.5.1.1*
2.5.2	*2.1.5.2 CM Service Provider Timers*
2.5.2.1	*2.1.5.2.1*
2.5.2.2	*2.1.5.2.2*
2.5.3	*2.1.5.3 CM-ASE Protocol Description*
2.5.3.1	*2.1.5.3.1 Introduction*
2.5.3.1.1	*2.1.5.3.1.1*
2.5.3.1.2	*2.1.5.3.1.2*
2.5.3.1.3	*2.1.5.3.1.3*
2.5.3.1.4	*2.1.5.3.1.4*
2.5.3.2	*2.1.5.3.2 CM-Air-ASE Protocol Description*
2.5.3.2.1	*2.1.5.3.2.1*
2.5.3.2.2	*2.1.5.3.2.2 D-START Indication*
2.5.3.2.2.1	*2.1.5.3.2.2.1*
2.5.3.2.2.1.1	*2.1.5.3.2.2.1.1*
2.5.3.2.2.1.2	*2.1.5.3.2.2.1.2*
2.5.3.2.2.1.3	*2.1.5.3.2.2.1.3*
2.5.3.2.3	*2.1.5.3.2.3 D-START Confirmation*
2.5.3.2.3.1	*2.1.5.3.2.3.1*
2.5.3.2.3.2	*2.1.5.3.2.3.2*
2.5.3.2.4	*2.1.5.3.2.4 D-DATA Indication*
2.5.3.2.4.1	*2.1.5.3.2.4.1*
2.5.3.2.4.1.1	*2.1.5.3.2.4.1.1*
2.5.3.2.4.1.2	*2.1.5.3.2.4.1.2*
2.5.3.2.4.1.3	*2.1.5.3.2.4.1.3*
2.5.3.2.4.2	*2.1.5.3.2.4.2*
2.5.3.2.4.2.1	*2.1.5.3.2.4.2.1*
2.5.3.2.4.2.2	*2.1.5.3.2.4.2.2*
2.5.3.2.4.2.3	*2.1.5.3.2.4.2.3*
2.5.3.2.5	*2.1.5.3.2.5 D-END Indication*
2.5.3.2.5.1	*2.1.5.3.2.5.1*
2.5.3.2.5.2	*2.1.5.3.2.5.2*
2.5.3.2.6	*2.1.5.3.2.6 CM-logon and CM-server-facility-query Service Request*
2.5.3.2.6.1	*2.1.5.3.2.6.1*
2.5.3.2.6.1.1	*2.1.5.3.2.6.1.1*
2.5.3.2.6.2	*2.1.5.3.2.6.2*
2.5.3.2.6.2.1	*2.1.5.3.2.6.2.1*
2.5.3.2.6.2.2	*2.1.5.3.2.6.2.2*

2.5.3.2.7	*2.1.5.3.2.7 CM-contact Service Response*
2.5.3.2.7.1	*2.1.5.3.2.7.1*
2.5.3.2.7.2	*2.1.5.3.2.7.2*
2.5.3.2.8	*2.1.5.3.2.8 CM-user-abort Service Request*
2.5.3.2.8.1	*2.1.5.3.2.8.1*
2.5.3.2.9	*2.1.5.3.2.9 D-ABORT Indication*
2.5.3.2.9.1	*2.1.5.3.2.9.1*
2.5.3.2.10	*2.1.5.3.2.10 D-P-ABORT Indication*
2.5.3.2.10.1	*2.1.5.3.2.10.1*
2.5.3.3	*2.1.5.3.3 CM-Ground-ASE Protocol Description*
2.5.3.3.1	*2.1.5.3.3.1*
2.5.3.3.2	*2.1.5.3.3.2 D-START Indication*
2.5.3.3.2.1	*2.1.5.3.3.2.1*
2.5.3.3.2.1.1	*2.1.5.3.3.2.1.1*
2.5.3.3.2.1.2	*2.1.5.3.3.2.1.2*
2.5.3.3.2.1.3	*2.1.5.3.3.2.1.3*
2.5.3.3.2.2	*2.1.5.3.3.2.2*
2.5.3.3.2.2.1	*2.1.5.3.3.2.2.1*
2.5.3.3.2.2.2	*2.1.5.3.3.2.2.2*
2.5.3.3.2.2.3	*2.1.5.3.3.2.2.3*
2.5.3.3.2.2.4	*2.1.5.3.3.2.2.4*
2.5.3.3.2.3	*2.1.5.3.3.2.3*
2.5.3.3.3	*2.1.5.3.3.3 D-START Confirmation*
2.5.3.3.3.1	*2.1.5.3.3.3.1*
2.5.3.3.3.1.1	*2.1.5.3.3.3.1.1*
2.5.3.3.3.1.2	*2.1.5.3.3.3.1.2*
2.5.3.3.3.1.3	*2.1.5.3.3.3.1.3*
2.5.3.3.4	*2.1.5.3.3.4 D-DATA Indication*
2.5.3.3.4.1	*2.1.5.3.3.4.1*
2.5.3.3.4.2	*2.1.5.3.3.4.2*
2.5.3.3.4.3	*2.1.5.3.3.4.3*
2.5.3.3.4.4	*2.1.5.3.3.4.4*
2.5.3.3.5	*2.1.5.3.3.5 D-END Confirmation*
2.5.3.3.5.1	*2.1.5.3.3.5.1*
2.5.3.3.6	*2.1.5.3.3.6 CM-logon and CM-server-facility-query Service Response*
2.5.3.3.6.1	*2.1.5.3.3.6.1*
2.5.3.3.6.2	*2.1.5.3.3.6.2*
2.5.3.3.6.3	*2.1.5.3.3.6.3*
2.5.3.3.6.3.1	*2.1.5.3.3.6.3.1*
2.5.3.3.6.3.2	*2.1.5.3.3.6.3.2*
2.5.3.3.7	*2.1.5.3.3.7 CM-update and CM-server-facility-update Service Request*
2.5.3.3.7.1	*2.1.5.3.3.7.1*
2.5.3.3.7.2	*2.1.5.3.3.7.2*
2.5.3.3.7.3	*2.1.5.3.3.7.3*
2.5.3.3.7.4	*2.1.5.3.3.7.4*
2.5.3.3.8	*2.1.5.3.3.8 CM-contact Service Request*
2.5.3.3.8.1	*2.1.5.3.3.8.1*
2.5.3.3.8.2	*2.1.5.3.3.8.2*
2.5.3.3.9	*2.1.5.3.3.9 CM-end Service Request*
2.5.3.3.9.1	*2.1.5.3.3.9.1*
2.5.3.3.10	*2.1.5.3.3.10 CM-forward Service Request*
2.5.3.3.10.1	*2.1.5.3.3.10.1*
2.5.3.3.11	*2.1.5.3.3.11 CM-user-abort Service Request*
2.5.3.3.11.1	*2.1.5.3.3.11.1*
2.5.3.3.12	*2.1.5.3.3.12 D-ABORT Indication*
2.5.3.3.12.1	*2.1.5.3.3.12.1*
2.5.3.3.13	*2.1.5.3.3.13 D-P-ABORT Indication*
2.5.3.3.13.1	*2.1.5.3.3.13.1*

2.5.4	*2.1.5.4 Exception Handling*
2.5.4.1	*2.1.5.4.1 Timer Expiration*
2.5.4.1.1	*2.1.5.4.1.1*
2.5.4.2	*2.1.5.4.2 Unrecoverable System Error*
2.5.4.2.1	*2.1.5.4.2.1*
2.5.4.3	*2.1.5.4.3 Invalid PDU*
2.5.4.3.1	*2.1.5.4.3.1*
2.5.4.3.2	*2.1.5.4.3.2*
2.5.4.4	*2.1.5.4.4 Not Permitted PDU or Dialogue Service Primitive*
2.5.4.4.1	*2.1.5.4.4.1*
2.5.4.4.2	*2.1.5.4.4.2*
2.5.4.4.3	*2.1.5.4.4.3*
2.5.4.5	*2.1.5.4.5 D-START Confirmation Result or Reject Source Parameter Values Not as Expected*
2.5.4.5.1	*2.1.5.4.5.1*
2.5.4.5.2	*2.1.5.4.5.2*
2.5.4.6	*2.1.5.4.6 D-END Confirmation Not as Expected*
2.5.4.6.1	*2.1.5.4.6.1*
2.5.4.7	*2.1.5.4.7 D-START Indication Quality of Service Parameter Not as Expected*
2.5.4.7.1	*2.1.5.4.7.1*
2.5.4.8	*2.1.5.4.8 Expected PDU Not Provided*
2.5.4.8.1	*2.1.5.4.8.1*
2.5.4.8.2	*2.1.5.4.8.2*
2.5.4.9	*2.1.5.4.9 D-START Security Requirements Parameter Not as Expected*
2.5.4.9.1	*2.1.5.4.9.1*
2.5.5	*2.1.5.5 CM ASE State Tables*
2.5.5.1	*2.1.5.5.1 Priority*
2.5.5.1.1	*2.1.5.5.1.1*
2.6	*2.1.6 COMMUNICATION REQUIREMENTS*
2.6.1	*2.1.6.1 Encoding Rules*
2.6.1.1	*2.1.6.1.1*
2.6.2	*2.1.6.2 Dialogue Service Requirements*
2.6.2.1	*2.1.6.2.1 Primitive Requirements*
2.6.2.1.1	*2.1.6.2.1.1*
2.6.2.2	*2.1.6.2.2 ATN Quality-of-Service Requirements*
2.6.2.2.1	*2.1.6.2.2.1*
2.6.2.2.2	*2.1.6.2.2.2*
2.6.2.2.3	*2.1.6.2.2.3*
2.6.2.3	*2.1.6.2.3 ATN Security Requirements*
2.6.2.3.1	*2.1.6.2.3.1*
2.6.2.3.2	*2.1.6.2.3.2*
2.6.2.3.3	
2.6.2.3.4	
2.7	*2.1.7 CM USER REQUIREMENTS*
2.7.1	*2.1.7.1 CM-Air-User Requirements*
2.7.1.1	*2.1.7.1.1 General CM-air-user Requirements*
2.7.1.1.1	*2.1.7.1.1.1*
2.7.1.1.1.1	*2.1.7.1.1.1.1*
2.7.1.2	*2.1.7.1.2 CM-logon Service Requirements*
2.7.1.2.1	*2.1.7.1.2.1*
2.7.1.2.2	*2.1.7.1.2.2*
2.7.1.2.3	*2.1.7.1.2.3*
2.7.1.2.4	*2.1.7.1.2.4*
2.7.1.2.5	*2.1.7.1.2.5*
2.7.1.2.6	*2.1.7.1.2.6*
2.7.1.2.7	*2.1.7.1.2.7*
2.7.1.2.8	*2.1.7.1.2.8*
2.7.1.2.9	*2.1.7.1.2.9*

2.7.1.2.10	*2.1.7.1.2.10*
2.7.1.2.11	*2.1.7.1.2.11*
2.7.1.3	*2.1.7.1.3 CM-update Service Requirements*
2.7.1.3.1	*2.1.7.1.3.1*
2.7.1.3.2	*2.1.7.1.3.2*
2.7.1.3.3	*2.1.7.1.3.3*
2.7.1.4	*2.1.7.1.4 CM-contact Service Requirements*
2.7.1.4.1	*2.1.7.1.4.1*
2.7.1.4.2	*2.1.7.1.4.2*
2.7.1.4.3	*2.1.7.1.4.3*
2.7.1.4.4	*2.1.7.1.4.4*
2.7.1.4.5	*2.1.7.1.4.5*
2.7.1.5	*2.1.7.1.5 CM-server-facility-query Service Requirements*
2.7.1.5.1	*2.1.7.1.5.1*
2.7.1.5.2	*2.1.7.1.5.2*
2.7.1.5.3	*2.1.7.1.5.3*
2.7.1.5.4	*2.1.7.1.5.4*
2.7.1.5.5	*2.1.7.1.5.5*
2.7.1.5.6	*2.1.7.1.5.6*
2.7.1.6	*2.1.7.1.6 CM-server-facility-update Service Requirements*
2.7.1.6.1	*2.1.7.1.6.1*
2.7.1.6.2	*2.1.7.1.6.2*
2.7.1.6.3	*2.1.7.1.6.3*
2.7.1.7	*2.1.7.1.7 CM-user-abort Service Requirements*
2.7.2	*2.1.7.2 CM-Ground-User Requirements*
2.7.2.1	*2.1.7.2.1 General CM-Ground-User Requirements.*
2.7.2.1.1	*2.1.7.2.1.1*
2.7.2.1.2	*2.1.7.2.1.2*
2.7.2.1.3	*2.1.7.2.1.3*
2.7.2.1.3.1	*2.1.7.2.1.3.1*
2.7.2.1.3.2	*2.1.7.2.1.3.2*
2.7.2.1.4	*2.1.7.2.1.4*
2.7.2.1.5	*2.1.7.2.1.5*
2.7.2.2	*2.1.7.2.2 CM-logon Service Requirements*
2.7.2.2.1	*2.1.7.2.2.1*
2.7.2.2.2	*2.1.7.2.2.2*
2.7.2.2.3	*2.1.7.2.2.3*
2.7.2.2.4	*2.1.7.2.2.4*
2.7.2.2.5	*2.1.7.2.2.5*
2.7.2.2.6	*2.1.7.2.2.6*
2.7.2.2.7	*2.1.7.2.2.7*
2.7.2.2.8	*2.1.7.2.2.8*
2.7.2.2.9	*2.1.7.2.2.9*
2.7.2.2.10	*2.1.7.2.2.10*
2.7.2.2.11	*2.1.7.2.2.11*
2.7.2.2.12	*2.1.7.2.2.12*
2.7.2.3	*2.1.7.2.3 CM-update Service Requirements*
2.7.2.3.1	*2.1.7.2.3.1*
2.7.2.3.2	*2.1.7.2.3.2*
2.7.2.3.3	*2.1.7.2.3.3*
2.7.2.3.4	*2.1.7.2.3.4*
2.7.2.4	*2.1.7.2.4 CM-contact Service Requirements*
2.7.2.4.1	*2.1.7.2.4.1*
2.7.2.5	*2.1.7.2.5 CM-end Service Requirements*
2.7.2.6	*2.1.7.2.6 CM-forward Service Requirements*
2.7.2.6.1	*2.1.7.2.6.1*
2.7.2.6.2	*2.1.7.2.6.2*
2.7.2.6.3	*2.1.7.2.6.3*

2.7.2.6.4	*2.1.7.2.6.4*
2.7.2.6.5	*2.1.7.2.6.5*
2.7.2.6.6	*2.1.7.2.6.6*
2.7.2.6.7	*2.1.7.2.6.7*
2.7.2.6.8	*2.1.7.2.6.8*
2.7.2.6.9	*2.1.7.2.6.9*
2.7.2.7	*2.1.7.2.7 CM-server-facility-query Service Requirements*
2.7.2.7.1	*2.1.7.2.7.1*
2.7.2.7.2	*2.1.7.2.7.2*
2.7.2.7.3	*2.1.7.2.7.3*
2.7.2.7.4	*2.1.7.2.7.4*
2.7.2.7.5	*2.1.7.2.7.5*
2.7.2.7.6	*2.1.7.2.7.6*
2.7.2.7.7	*2.1.7.2.7.7*
2.7.2.7.8	*2.1.7.2.7.8*
2.7.2.7.9	*2.1.7.2.7.9*
2.7.2.8	*2.1.7.2.8 CM-server-facility-update Service Requirements*
2.7.2.8.1	*2.1.7.2.8.1*
2.7.2.8.2	*2.1.7.2.8.2*
2.7.2.8.3	*2.1.7.2.8.3*
2.7.2.9	*2.1.7.2.9 CM-user-abort Service Requirements*
2.7.3	*2.1.7.3 Parameter Value Unit, Range and Resolution*
2.7.3.1	*2.1.7.3.1*
2.8	*2.1.8 SUBSETTING RULES*
2.8.1	*2.1.8.1 General*
2.8.1.1	*2.1.8.1.1*
END	*END*

Doc 9880 (1st edition)	**Paragraph numbering as in Doc 9705 – third edition (2002)**
Chapter 3	**2.3 CONTROLLER PILOT DATA LINK COMMUNICATION APPLICATION**
	2.3.1 INTRODUCTION
1.5.1 and 1.5.4	2.3.1.1 Overview
1.5.2	2.3.1.1.1
1.5.3	2.3.1.1.2
3.1	2.3.1.1.3
3.1.1	
3.1.2	*Note 1.— Structure:*
3.1.2.1	*Note 2.— Functional Descriptions*
3.1.2.2	*a) Controller-pilot message exchange function*
3.1.2.3	*b) Transfer of data authority function*
3.1.2.4	*c) Down stream clearance function*
	d) Ground forward function
3.2	*Note 3.*
3.2.1	2.3.2 GENERAL REQUIREMENTS
3.2.1.1	2.3.2.1 CPDLC ASE Version Number
3.2.2	2.3.2.1.1
3.2.2.1	2.3.2.2 Error Processing Requirements
3.2.2.2	2.3.2.2.1
3.3	2.3.2.2.2
3.3.1	2.3.3 THE ABSTRACT SERVICE
3.3.1.1	2.3.3.1 Service Description
3.3.2	2.3.3.1.1
3.3.2.1	2.3.3.2 The CPDLC-ASE Abstract Service
3.3.3	2.3.3.2.1
3.3.3.1	2.3.3.3 CPDLC-start Service

3.3.3.2	2.3.3.3.1
3.3.3.2.1	2.3.3.3.2 Called peer identifier
3.3.3.2.2.	2.3.3.3.2.1
3.3.3.3	2.3.3.3.2.2
3.3.3.3.1	2.3.3.3.3 Calling peer identifier
3.3.3.3.2	2.3.3.3.3.1
3.3.3.4	2.3.3.3.3.2
3.3.3.4.1	2.3.3.3.4 CPDLC Message
3.3.3.4.2	2.3.3.3.4.1
3.3.3.5	2.3.3.3.4.2
3.3.3.5.1	2.3.3.3.5 Reject Reason
3.3.3.5.2	2.3.3.3.5.1
	2.3.3.3.5.2
3.3.3.6	2.3.3.3.5.3
3.3.3.6.1	2.3.3.3.6 Result
3.3.3.7	2.3.3.3.6.1
3.3.3.7.1	2.3.3.3.7 Class of Communication Service
3.3.3.7.2	2.3.3.3.7.1
3.3.3.8	2.3.3.3.7.2
3.3.3.8.1	2.3.3.3.8 Security Required
	2.3.3.3.8.1
3.3.4	2.3.3.3.8.2
3.3.4.1	2.3.3.4 DSC-start Service
3.3.4.2	2.3.3.4.1
3.3.4.2.1	2.3.3.4.2
3.3.4.3	2.3.3.4.2.1
3.3.4.3.1	2.3.3.4.3 Aircraft Address
3.3.4.4	2.3.3.4.3.1
3.3.4.4.1	2.3.3.4.4 CPDLC Message
3.3.4.5	2.3.3.4.4.1
3.3.4.5.1	2.3.3.4.5 Reject Reason
	2.3.3.4.5.1
3.3.4.6	2.3.3.4.5.2
3.3.4.6.1	2.3.3.4.6 Result
3.3.4.7	2.3.3.4.6.1
3.3.4.7.1	2.3.3.4.7 Class of Communication Service
3.3.4.7.2	2.3.3.4.7.1
3.3.4.8	2.3.3.4.7.2
3.3.4.8.1	2.3.3.4.8 Security Required
	2.3.3.4.8.1
3.3.5	2.3.3.4.8.2
3.3.5.1	2.3.3.5 CPDLC-message Service
3.3.5.2	2.3.3.5.1
3.3.5.2.1	2.3.3.5.2 CPDLC Message
3.3.5.2.2	2.3.3.5.2.1
3.3.6	2.3.3.5.2.2
3.3.6.1	2.3.3.6 CPDLC-end Service
3.3.6.2	2.3.3.6.1
3.3.6.2.1	2.3.3.6.2 CPDLC Message
3.3.6.2.2	2.3.3.6.2.1
3.3.6.3	2.3.3.6.2.2
3.3.6.3.1	2.3.3.6.3 Result
3.3.7	2.3.3.6.3.1
3.3.7.1	2.3.3.7 DSC-end Service
3.3.7.2	2.3.3.7.1
3.3.7.2.1	2.3.3.7.2 CPDLC message
3.3.7.2.2	2.3.3.7.2.1
3.3.7.3	2.3.3.7.2.2

3.3.7.3.1	2.3.3.7.3 Result
3.3.8	2.3.3.7.3.1
3.3.8.1	2.3.3.8 CPDLC-forward Service
3.3.8.2	2.3.3.8.1
3.3.8.3	2.3.3.8.2
3.3.8.3.1	2.3.3.8.3 Called Facility Designation
3.3.8.4	2.3.3.8.3.1
3.3.8.4.1	2.3.3.8.4 Calling Facility Designation
3.3.8.5	2.3.3.8.4.1
3.3.8.5.1	2.3.3.8.5 ASE Version Number
3.3.8.5.2	2.3.3.8.5.1
3.3.8.6	2.3.3.8.5.2
3.3.8.6.1	2.3.3.8.6 CPDLC Message
3.3.8.7	2.3.3.8.6.1
3.3.8.7.1	2.3.3.8.7 Class of Communication Service
3.3.8.8	2.3.3.8.7.1
3.3.8.8.1	2.3.3.8.8 Result
3.3.8.9	2.3.3.8.8.1
3.3.8.9.1	2.3.3.8.9 Security Required
	2.3.3.8.9.1
	2.3.3.8.9.2
	2.3.3.8.10 Emulated Version
3.3.9	2.3.3.8.10.1
3.3.9.1	2.3.3.9 CPDLC-user-abort Service
3.3.9.2	2.3.3.9.1
3.3.9.2.1	2.3.3.9.2 Reason
3.3.9.2.2	2.3.3.9.2.1
3.3.10	2.3.3.9.2.2
3.3.10.1	2.3.3.10 CPDLC-provider-abort Service
3.3.10.2	2.3.3.10.1
3.3.10.2.1	2.3.3.10.2 Reason
3.4	2.3.3.10.2.1
3.4.1	2.3.4 FORMAL DEFINITIONS OF MESSAGES
3.4.1.1	2.3.4.1 Encoding/Decoding Rules
3.4.1.2	2.3.4.1.1
3.4.1.3	2.3.4.1.2
3.4.1.4	
3.4.1.5	
3.4.2	
3.4.2.1	2.3.4.2 CPDLC ASN.1 Abstract Syntax
3.4.3	2.3.4.2.1
3.4.3.1	
3.5	
3.5.1	2.3.5 PROTOCOL DEFINITION
3.5.1.1	2.3.5.1 Sequence Rules
3.5.2	2.3.5.1.1
3.5.2.1	2.3.5.2 CPDLC Service Provider Timers
3.5.2.2	2.3.5.2.1
3.5.3	2.3.5.2.2
3.5.3.1	2.3.5.3 CPDLC-Air-ASE Protocol Description
3.5.3.1.1	2.3.5.3.1 Introduction
3.5.3.1.2	2.3.5.3.1.1
3.5.3.1.3	2.3.5.3.1.2
3.5.3.1.4	2.3.5.3.1.3
3.5.3.1.5	2.3.5.3.1.4
3.5.3.1.6	2.3.5.3.1.5
3.5.3.2	2.3.5.3.1.6
3.5.3.2.1	2.3.5.3.2 D-START Indication

3.5.3.3	2.3.5.3.2.1
3.5.3.3.1	2.3.5.3.3 D-START Confirmation
3.5.3.3.2	2.3.5.3.3.1
3.5.3.3.3	2.3.5.3.3.2
3.5.3.3.4	2.3.5.3.3.3
3.5.3.4	2.3.5.3.3.4
3.5.3.4.1	2.3.5.3.4 D-DATA Indication
3.5.3.4.2	2.3.5.3.4.1
3.5.3.5	2.3.5.3.4.2
3.5.3.5.1	2.3.5.3.5 D-END Indication
3.5.3.6	2.3.5.3.5.1
3.5.3.6.1	2.3.5.3.6 D-END Confirmation
3.5.3.6.2	2.3.5.3.6.1
3.5.3.7	2.3.5.3.6.2
3.5.3.7.1	2.3.5.3.7 CPDLC-start Service Request
3.5.3.8	2.3.5.3.7.1
3.5.3.8.1	2.3.5.3.8 CPDLC-start service response
3.5.3.8.2	2.3.5.3.8.1
3.5.3.9	2.3.5.3.8.2
3.5.3.9.1	2.3.5.3.9 DSC-start Service Request
3.5.3.10	2.3.5.3.9.1
3.5.3.10.1	2.3.5.3.10 CPDLC-message Service Request
3.5.3.10.2	2.3.5.3.10.1
3.5.3.11	2.3.5.3.10.2
3.5.3.11.1	2.3.5.3.11 CPDLC-end Service Response
3.5.3.11.2	2.3.5.3.11.1
3.5.3.12	2.3.5.3.11.2
3.5.3.12.1	2.3.5.3.12 DSC-end Service Request
3.5.3.13	2.3.5.3.12.1
3.5.3.13.1	2.3.5.3.13 CPDLC-user-abort Service Request
3.5.3.14	2.3.5.3.13.1
3.5.3.14.1	2.3.5.3.14 D-ABORT Indication
3.5.3.14.2	2.3.5.3.14.1
3.5.3.15	2.3.5.3.14.2
3.5.3.15.1	2.3.5.3.15 D-P-ABORT Indication
3.5.4	2.3.5.3.15.1
3.5.4.1	2.3.5.4 CPDLC-Air-ASE Exception Handling
3.5.4.1.1	2.3.5.4.1 A Timer Expires
3.5.4.2	2.3.5.4.1.1
3.5.4.2.1	2.3.5.4.2 Unrecoverable System Error
3.5.4.3	2.3.5.4.2.1
3.5.4.3.1	2.3.5.4.3 Invalid PDU
3.5.4.3.2	2.3.5.4.3.1
3.5.4.4	2.3.5.4.3.2
3.5.4.4.1	2.3.5.4.4 Protocol Error
3.5.4.4.2	2.3.5.4.4.1
3.5.4.4.3	2.3.5.4.4.2
3.5.4.4.4	2.3.5.4.4.3
3.5.4.5	2.3.5.4.4.4
3.5.4.5.1	2.3.5.4.5 D-START Confirmation Result or Reject Source Not as Expected
3.5.4.6	2.3.5.4.5.1
3.5.4.6.1	2.3.5.4.6 D-START Indication Quality of Service Not as Expected
3.5.4.7	2.3.5.4.6.1
3.5.4.7.1	2.3.5.4.7 Expected PDU Missing
3.5.4.8	2.3.5.4.7.1
3.5.4.8.1	2.3.5.4.8 D-START Security Requirements Parameter Not as Expected
3.5.5	2.3.5.4.8.1
3.5.5.1	2.3.5.5 CPDLC-Ground-ASE Protocol Description

3.5.5.1.1	2.3.5.5.1 Introduction
3.5.5.1.2	2.3.5.5.1.1
3.5.5.1.3	2.3.5.5.1.2
3.5.5.1.4	2.3.5.5.1.3
3.5.5.1.5	2.3.5.5.1.4
3.5.5.1.6	2.3.5.5.1.5
3.5.5.2	2.3.5.5.1.6
3.5.5.2.1	2.3.5.5.2 D-START Indication
3.5.5.2.2	2.3.5.5.2.1
3.5.5.2.3	2.3.5.5.2.2
3.5.5.2.4	2.3.5.5.2.3
3.5.5.3	2.3.5.5.2.4
3.5.5.3.1	2.3.5.5.3 D-START Confirmation
3.5.5.3.2	2.3.5.5.3.1
3.5.5.3.3	2.3.5.5.3.2
3.5.5.4	2.3.5.5.3.3
3.5.5.4.1	2.3.5.5.4 D-DATA Indication
3.5.5.4.2	2.3.5.5.4.1
3.5.5.5	2.3.5.5.4.2
3.5.5.5.1	2.3.5.5.5 D-END Indication
3.5.5.6	2.3.5.5.5.1
3.5.5.6.1	2.3.5.5.6 D-END Confirmation
3.5.5.6.2	2.3.5.5.6.1
3.5.5.7	2.3.5.5.6.2
3.5.5.7.1	2.3.5.5.7 CPDLC-start Service Request
3.5.5.8	2.3.5.5.7.1
3.5.5.8.1	2.3.5.5.8 CPDLC-start Service Response
3.5.5.8.2	2.3.5.5.8.1
3.5.5.9	2.3.5.5.8.2
3.5.5.9.1	2.3.5.5.9 DSC-start Service Response
3.5.5.9.2	2.3.5.5.9.1
3.5.5.10	2.3.5.5.9.2
3.5.5.10.1	2.3.5.5.10 CPDLC-message Service Request
3.5.5.10.2	2.3.5.5.10.1
3.5.5.11	2.3.5.5.10.2
3.5.5.11.1	2.3.5.5.11 CPDLC-end Service Request
3.5.5.12	2.3.5.5.11.1
3.5.5.12.1	2.3.5.5.12 DSC-end Service Response
3.5.5.12.2	2.3.5.5.12.1
3.5.5.13	2.3.5.5.12.2
3.5.5.13.1	2.3.5.5.13 CPDLC-forward Service Request
3.5.5.14	2.3.5.5.13.1
3.5.5.14.1	2.3.5.5.14 CPDLC-user-abort Service Request
3.5.5.15	2.3.5.5.14.1
3.5.5.15.1	2.3.5.5.15 D-ABORT Indication
3.5.5.15.2	2.3.5.5.15.1
3.5.5.16	2.3.5.5.15.2
3.5.5.16.1	2.3.5.5.16 D-P-ABORT Indication
3.5.6	2.3.5.5.16.1
3.5.6.1	2.3.5.6 CPDLC-Ground-ASE Exception Handling
3.5.6.1.1	2.3.5.6.1 A Timer Expires
3.5.6.2	2.3.5.6.1.1
3.5.6.2.1	2.3.5.6.2 Unrecoverable System Error
3.5.6.3	2.3.5.6.2.1
3.5.6.3.1	2.3.5.6.3 Invalid PDU
3.5.6.3.2	2.3.5.6.3.1
3.5.6.4	2.3.5.6.3.2
3.5.6.4.1	2.3.5.6.4 Protocol Error

3.5.6.4.2	2.3.5.6.4.1
3.5.6.4.3	2.3.5.6.4.2
3.5.6.4.4	2.3.5.6.4.3
3.5.6.5	2.3.5.6.4.4
3.5.6.5.1	2.3.5.6.5 D-START Confirmation Result or Reject Source Not as Expected
3.5.6.6	2.3.5.6.5.1
3.5.6.6.1	2.3.5.6.6 D-START Indication Quality of Service Not as Expected
3.5.6.7	2.3.5.6.6.1
3.5.6.7.1	2.3.5.6.7 Expected PDU Missing
3.5.6.8	2.3.5.6.7.1
3.5.6.8.1	2.3.5.6.8 D-START Security Requirements Parameter Not as Expected
3.5.7	2.3.5.6.8.1
3.5.7.1	2.3.5.7 CPDLC ASE State Tables
3.5.7.1.1	2.3.5.7.1 Priority
3.6	2.3.5.7.1.1
3.6.1	2.3.6 COMMUNICATION REQUIREMENTS
3.6.1.1	2.3.6.1 Encoding Rules
3.6.2	2.3.6.1.1
3.6.2.1	2.3.6.2 Dialogue Service Requirements
3.6.2.1.1	2.3.6.2.1 Primitive Requirements
3.6.2.2	2.3.6.2.1.1
3.6.2.2.1	2.3.6.2.2 Quality-of-Service Requirements
3.6.2.2.2	2.3.6.2.2.1
3.6.2.2.3	2.3.6.2.2.2
3.6.2.3	2.3.6.2.2.3
3.6.2.3.1	2.3.6.2.3 ATN Security Requirements
3.7	2.3.6.2.3.1
3.7.1	2.3.7 CPDLC USER REQUIREMENTS
3.7.1.1	2.3.7.1 General
3.7.1.1.1	2.3.7.1.1 General CPDLC Service Requirements
3.7.1.1.2	2.3.7.1.1.1
3.7.2	2.3.7.1.1.2
3.7.2.1	
3.7.2.1.1	
3.7.2.1.2	
3.7.2.1.3	
3.7.2.1.4	
3.7.2.1.5	
3.7.2.1.6	
3.7.2.1.7	
3.7.2.2	
3.7.2.2.1	
3.7.2.2.2	
3.7.2.2.3	
3.7.2.2.4	
3.7.2.2.5	
3.7.2.2.6	
3.7.3	
3.7.3.1	
3.7.3.2	
3.7.3.3	
3.7.3.3.1	
3.7.3.3.2	
3.7.3.3.3	
3.7.4	
3.7.4.1	2.3.7.2 CPDLC Message Generation Requirements
3.7.4.1.1	2.3.7.2.1 Message Composition
3.7.4.1.2	2.3.7.2.1.1

3.7.4.1.3	2.3.7.2.1.2
3.7.4.1.4	2.3.7.2.1.3
3.7.4.1.5	2.3.7.2.1.4
3.7.4.1.6	2.3.7.2.1.5
3.7.4.2	2.3.7.2.1.6
3.7.4.2.1	2.3.7.2.2 Message Identification Number
3.7.4.2.2	2.3.7.2.2.1
3.7.4.2.3	2.3.7.2.2.2
3.7.4.3	2.3.7.2.2.3
3.7.4.3.1	2.3.7.2.3 Message Elements That Cannot Be Combined With Other Message Elements in a Message
3.7.4.3.2	2.3.7.2.3.1
3.7.4.4	2.3.7.2.3.2
3.7.4.4.1	2.3.7.2.4 Restriction on Route Clearance Variable Message Elements
3.7.5	2.3.7.2.4.1
3.7.5.1	
3.7.5.1.1	
3.7.5.1.2	
3.7.5.1.3	
3.7.5.1.4	
3.7.5.1.5	
3.7.5.1.6	
3.7.5.1.7	
3.7.5.2	
3.7.5.2.1	
3.7.5.2.2	
3.7.5.2.3	
3.7.5.2.4	
3.7.5.2.5	
3.7.5.2.6	
3.7.5.2.7	
3.7.6	
3.7.6.1	2.3.7.3 CPDLC Message Receipt Requirements
3.7.6.1.1	2.3.7.3.1 Message Attributes
3.7.6.1.2	2.3.7.3.1.1
3.7.6.2	2.3.7.3.1.2
3.7.6.2.1	2.3.7.3.2 Urgency Requirements
3.7.6.2.2	2.3.7.3.2.1
3.7.6.2.3	2.3.7.3.2.2
3.7.6.3	2.3.7.3.2.3
3.7.6.3.1	2.3.7.3.3 Alerting Requirements
3.7.6.3.2	2.3.7.3.3.1
3.7.6.4	2.3.7.3.3.2
3.7.6.4.1	2.3.7.3.4 Response Attribute
3.7.6.4.2	2.3.7.3.4.1
3.7.6.5	2.3.7.3.4.2
3.7.6.5.1	2.3.7.3.5 CPDLC/DSC Distinction
3.7.6.6	2.3.7.3.5.1
3.7.6.6.1	2.3.7.3.6 Air/Ground - Ground/Ground Distinction
3.7.6.7	2.3.7.3.6.1
3.7.6.7.1	2.3.7.3.7 Logical Acknowledgment Prohibited
3.7.6.7.2	2.3.7.3.7.1
3.7.6.8	2.3.7.3.7.2
3.7.6.8.1	2.3.7.3.8 Message Reference Numbers
3.7.6.8.2	2.3.7.3.8.1
3.7.6.9	2.3.7.3.8.2
3.7.6.9.1	2.3.7.3.9 Message Response Requirements
3.7.6.9.2	2.3.7.3.9.1

3.7.6.9.3	2.3.7.3.9.2
3.7.6.9.4	2.3.7.3.9.3
3.7.6.9.5	2.3.7.3.9.4
3.7.6.9.6	2.3.7.3.9.5
3.7.6.9.7	2.3.7.3.9.6
3.7.6.9.8	2.3.7.3.9.7
3.7.6.9.9	2.3.7.3.9.8
3.7.6.9.10	2.3.7.3.9.9
3.7.6.9.11	2.3.7.3.9.10
3.7.6.9.12	2.3.7.3.9.11
3.7.6.9.13	2.3.7.3.9.12
3.7.6.9.14	2.3.7.3.9.13
3.7.6.9.15	2.3.7.3.9.14
3.7.6.9.16	2.3.7.3.9.15
3.7.6.9.17	2.3.7.3.9.16
3.7.6.9.18	2.3.7.3.9.17
3.7.6.9.19	2.3.7.3.9.18
3.7.6.9.20	2.3.7.3.9.19
3.7.6.9.21	2.3.7.3.9.20
3.7.6.9.22	2.3.7.3.9.21
3.7.6.10	2.3.7.3.9.22
3.7.6.10.1	2.3.7.3.10 Invalid Message Elements
3.7.6.11	2.3.7.3.10.1
3.7.6.11.1	2.3.7.3.11 Error Conditions
3.7.6.11.1.1	2.3.7.3.11.1 Duplicate Message Identification Numbers
3.7.6.11.2	2.3.7.3.11.1.1
3.7.6.11.2.1	2.3.7.3.11.2 Invalid Reference Number
3.7.6.11.3	2.3.7.3.11.2.1
3.7.6.11.3.1	2.3.7.3.11.3 No Available Message Identification Numbers
3.7.6.11.4	2.3.7.3.11.3.1
3.7.6.11.4.1	2.3.7.3.11.4 Insufficient Resources
3.7.6.11.5	2.3.7.3.11.4.1
3.7.6.11.5.1	2.3.7.3.11.5 Invalid Message Element Combination
3.7.6.11.6	2.3.7.3.11.5.1
3.7.6.11.6.1	2.3.7.3.11.6 Invalid Message Elements
3.7.6.11.7	2.3.7.3.11.6.1
3.7.6.11.7.1	2.3.7.3.11.7 System Management Responses
3.7.6.11.7.2	2.3.7.3.11.7.1
3.7.6.11.7.3	2.3.7.3.11.7.2
3.7.6.11.7.4	2.3.7.3.11.7.3
3.7.6.11.8	2.3.7.3.11.7.4
3.7.6.11.8.1	2.3.7.3.11.8 Invalid Message Response
3.7.6.11.8.2	2.3.7.3.11.8.1
3.7.6.11.8.3	2.3.7.3.11.8.2
3.7.6.11.8.4	2.3.7.3.11.8.3
3.7.6.11.8.5	2.3.7.3.11.8.4
3.7.6.11.9	2.3.7.3.11.8.5
3.7.6.11.9.1	2.3.7.3.11.9 Backward compatibility
3.7.6.11.10	2.3.7.3.11.9.1
3.7.6.11.10.1	
3.7.6.11.10.2	
3.7.7	
3.7.7.1	2.3.7.4 CPDLC-air-user Requirements
3.7.7.1.1	2.3.7.4.1 The CPDLC-start Service
3.7.7.1.1.1	2.3.7.4.1.1 Invoking the CPDLC-start request
3.7.7.1.1.2	2.3.7.4.1.1.1
3.7.7.1.1.3	2.3.7.4.1.1.2
3.7.7.1.2	2.3.7.4.1.1.3

3.7.7.1.2.1	2.3.7.4.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response
3.7.7.1.2.2	2.3.7.4.1.2.1
3.7.7.1.2.3	2.3.7.4.1.2.2
3.7.7.1.2.4	2.3.7.4.1.2.3
3.7.7.1.2.5	2.3.7.4.1.2.4
3.7.7.1.2.6	2.3.7.4.1.2.5
3.7.7.1.2.7	2.3.7.4.1.2.6
3.7.7.1.2.8	2.3.7.4.1.2.7
3.7.7.1.3	2.3.7.4.1.2.8
3.7.7.1.3.1	2.3.7.4.1.3 Receipt of a CPDLC-start confirmation
3.7.7.1.3.2	2.3.7.4.1.3.1
3.7.7.2	
3.7.7.2.1	2.3.7.4.2 The DSC-start Service
3.7.7.2.1.1	2.3.7.4.2.1 Invoking the DSC-start request
3.7.7.2.1.2	2.3.7.4.2.1.1
3.7.7.2.1.3	2.3.7.4.2.1.2
3.7.7.2.2	2.3.7.4.2.1.3
3.7.7.2.2.1	2.3.7.4.2.2 Receipt of a DSC-start confirmation
3.7.7.3	2.3.7.4.2.2.1
3.7.7.3.1	2.3.7.4.3 The CPDLC-message Service
3.7.7.3.1.1	
3.7.7.3.2	
3.7.7.3.2.1	2.3.7.4.3.1 Receipt of a CPDLC-message Indication
3.7.7.3.2.2	2.3.7.4.3.1.1
3.7.7.3.2.3	2.3.7.4.3.1.2
3.7.7.3.2.4	2.3.7.4.3.1.3
3.7.7.3.2.5	2.3.7.4.3.1.4
3.7.7.3.2.6	2.3.7.4.3.1.5
3.7.7.4	
3.7.7.4.1	2.3.7.4.4 The CPDLC-end Service
3.7.7.4.1.1	2.3.7.4.4.1 The CPDLC-end Service Request
3.7.7.4.2	2.3.7.4.4.1.1
3.7.7.4.2.1	2.3.7.4.4.2 Receipt of a CPDLC-end Indication and Invoking a CPDLC-end Response
3.7.7.4.2.2	2.3.7.4.4.2.1
3.7.7.4.2.2.1	2.3.7.4.4.2.2 Uplink Message Contains Error
3.7.7.4.2.3	2.3.7.4.4.2.2.1
3.7.7.4.2.3.1	2.3.7.4.4.2.3 No Message in the CPDLC-end Indication
3.7.7.4.2.3.2	2.3.7.4.4.2.3.1
3.7.7.4.2.4	2.3.7.4.4.2.3.2
3.7.7.4.2.4.1	2.3.7.4.4.2.4 Message in CPDLC-end Indication Does Not Require Response
3.7.7.4.2.4.2	2.3.7.4.4.2.4.1
3.7.7.4.2.5	2.3.7.4.4.2.4.2
3.7.7.4.2.5.1	2.3.7.4.4.2.5 Message in CPDLC-end Indication Requires Only Logical Acknowledgment
3.7.7.4.2.5.2	2.3.7.4.4.2.5.1
3.7.7.4.2.6	2.3.7.4.4.2.5.2
3.7.7.4.2.6.1	2.3.7.4.4.2.6 Message in CPDLC-end Indication With W/U, A/N, or R Attribute, Positive Response
3.7.7.4.2.7	2.3.7.4.4.2.6.1
3.7.7.4.2.7.1	2.3.7.4.4.2.7 Message in CPDLC-end Indication With W/U, A/N, or R Attribute, Negative Response
3.7.7.4.2.8	2.3.7.4.4.2.7.1
3.7.7.4.2.8.1	2.3.7.4.4.2.8 Message in CPDLC-end Indication With Y Response Attribute
3.7.7.4.2.8.2	2.3.7.4.4.2.8.1
3.7.7.4.2.9	2.3.7.4.4.2.8.2
3.7.7.4.2.10	2.3.7.4.4.2.9

3.7.7.5	2.3.7.4.4.2.10
3.7.7.5.1	2.3.7.4.5 The DSC-end Service
3.7.7.5.1.1	2.3.7.4.5.1 The DSC-end Request
3.7.7.5.1.2	2.3.7.4.5.1.1
3.7.7.6	2.3.7.4.5.1.2
3.7.7.6.1	2.3.7.4.6 The CPDLC-user-abort Service
3.7.7.6.1.1	2.3.7.4.6.1 Issuing a CPDLC-user-abort Request [commanded-termination]
3.7.7.6.2	2.3.7.4.6.1.1
3.7.7.6.2.1	2.3.7.4.6.2 Invoking a CPDLC-user-abort Request
3.7.7.6.2.2	2.3.7.4.6.2.1
3.7.7.6.3	2.3.7.4.6.2.2
3.7.7.6.3.1	2.3.7.4.6.3 Receipt of a CPDLC-abort Indication
3.7.7.6.3.2	2.3.7.4.6.3.1
3.7.8	2.3.7.4.6.3.2
3.7.8.1	2.3.7.5 CPDLC-Ground-User Requirements
3.7.8.1.1	2.3.7.5.1 The CPDLC-start Service
3.7.8.1.1.1	2.3.7.5.1.1 Invoking the CPDLC-start request
3.7.8.1.1.2	2.3.7.5.1.1.1
3.7.8.1.2	2.3.7.5.1.1.2
3.7.8.1.2.1	2.3.7.5.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response
3.7.8.1.2.2	2.3.7.5.1.2.1
3.7.8.1.2.3	2.3.7.5.1.2.2
3.7.8.1.2.4	2.3.7.5.1.2.3
3.7.8.1.2.5	2.3.7.5.1.2.4
3.7.8.1.2.6	2.3.7.5.1.2.5
3.7.8.1.2.7	
3.7.8.1.2.8	2.3.7.5.1.2.6
3.7.1.8.2.9	2.3.7.5.1.2.7
3.7.8.1.3	2.3.7.5.1.2.8
3.7.8.1.3.1	2.3.7.5.1.3 Receipt of a CPDLC-start confirmation
3.7.8.1.3.2	2.3.7.5.1.3.1
3.7.8.2	
3.7.8.2.1	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
3.7.8.2.1.1	2.3.7.5.2.1.1
3.7.8.2.1.2	2.3.7.5.2.1.2
3.7.8.2.1.3	2.3.7.5.2.1.3
3.7.8.2.1.4	2.3.7.5.2.1.4
3.7.8.2.1.5	2.3.7.5.2.1.5
3.7.8.2.1.6	2.3.7.5.2.1.6
3.7.8.2.1.7	2.3.7.5.2.1.7
3.7.8.2.2	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
3.7.8.2.1	2.3.7.5.2.2.2
3.7.8.2.2.2	
3.7.8.3	
3.7.8.3.1	
3.7.8.3.1.1	
3.7.8.3.2	
3.7.8.3.2.1	2.3.7.5.3 The CPDLC-end Service
3.7.8.4	2.3.7.5.3.1 The CPDLC-end Request
3.7.8.4.1	2.3.7.5.3.1.1
3.7.8.4.1.1	2.3.7.5.3.1.2
3.7.8.4.1.2	2.3.7.5.4 The DSC-end Service
3.7.8.5	2.3.7.5.4.1 Receipt of a DSC-end Indication and Invoking DSC-end Response
3.7.8.5.1	2.3.7.5.4.1.1 Downlink Message Contains Error
3.7.8.5.1.1	2.3.7.5.4.1.1.1

3.7.8.5.1.1.1	2.3.7.5.4.1.2 No Message in the DSC-end Indication
3.7.8.5.1.2	2.3.7.5.4.1.2.1
3.7.8.5.1.2.1	2.3.7.5.4.1.2.2
3.7.8.5.1.2.2	2.3.7.5.4.1.3 Message in DSC-end Indication Does Not Require Response
3.7.8.5.1.3	2.3.7.5.4.1.3.1
3.7.8.5.1.3.1	2.3.7.5.4.1.3.2
3.7.8.5.1.3.2	2.3.7.5.4.1.4 Message in DSC-end Indication Requires Only Logical
3.7.8.5.1.4	Acknowledgment
	2.3.7.5.4.1.4.1
3.7.8.5.1.4.1	2.3.7.5.4.1.4.2
3.7.8.5.1.4.2	2.3.7.5.4.1.5 Message in DSC-end Indication With Y Response Attribute
3.7.8.5.1.5	2.3.7.5.4.1.5.1
3.7.8.5.1.5.1	2.3.7.5.4.1.5.2
3.7.8.5.1.5.2	2.3.7.5.5 The CPDLC-forward Service
3.7.8.6	2.3.7.5.5.1 Invoking the CPDLC-forward Request
3.7.8.6.1	2.3.7.5.5.1.1
3.7.8.6.1.1	
3.7.8.6.1.2	2.3.7.5.5.2
	2.3.7.5.6 The CPDLC-user-abort Service
3.7.8.7	2.3.7.5.6.1 Issuing a CPDLC-user-abort Request
3.7.8.7.1	2.3.7.5.6.1.1
3.7.8.7.1.1	2.3.7.6 Message Intent
3.7.9	2.3.7.6.1 Purpose
3.7.9.1	2.3.7.6.2
3.7.9.2	2.3.7.6.3
	2.3.7.7 Parameter Value Unit, Range, and Resolution
3.7.10	2.3.7.7.1
3.7.710.1	2.3.8 SUBSETTING RULES
3.8	2.3.8.1 General
3.8.1	2.3.8.1.1
3.8.1.1	
	END
END	

Doc 9880 (1st edition)	**Paragraph numbering as in Doc 9705 – third edition (2002)**
Chapter 4	**2.4 FLIGHT INFORMATION SERVICES (in preparation)**

Doc 9880 (1st edition)	**Paragraph numbering as in Doc 9705 – third edition (2002)**
Chapter 5	**2.2 AUTOMATIC DEPENDENT SURVEILLANCE – BROADCAST (in preparation)**

Doc 9880 (1st edition)	**Paragraph numbering as in Doc 9705 – third edition (2002)**
Chapter 6	*This material is not included in Doc 9705*
6.1	
6.1.1	
6.1.1.1	
6.1.2	
6.1.2.1	
6.1.2.2	
6.1.2.2.1	
6.1.2.2.2	

6.1.2.3 6.1.2.3.1 6.1.2.3.2 6.1.2.4 6.1.2.4.1 **END**	

— END —