ACP-WGM11-WP05_CPDLC
29/05/06

International Civil Aviation Organization

WORKING PAPER

AERONAUTICAL COMMUNICATIONS PANEL (ACP)

11th Working Group M meeting

Montreal, Canada 31 May – 02 June 2006

Agenda Item 7: ATN Maintenance, Review of amendments to Doc. 9705 (source: ACP – WG-Whole)

TITLE

Prepared by EUROCONTROL

Presented by Nikos Fistas EUROCONTROL

SUMMARY

This paper provides the proposed revised text for Doc 9705 covering the introduction of the amended CPDLC application. The amendments were approved in ACP/WGW and were proposed to address risks that were identified by a safety assessment of the operational deployment of CPDLC applications in Europe in the frame of the EUROCONTROL LINK2000+ Programme.
The amended CPDLC mitigates the risk of undetected mis-delivery of messages to the end user and the risk of delivery of undetected corrupt messages to the end user. The proposed solution also reduces the integrity and availability requirements on information provided by the Context Management (CM) application and distributed across different systems during flight.

ACTION

To approve the proposed amendments of Doc 9705 in relation to the CPDLC application.

1. INTRODUCTION

1.1 This paper provides the revised text for Doc 9705 covering the introduction of the amended CPDLC application (previously referred to as PM-CPDLC) to address risks that were identified by a safety assessment of the operational deployment of CPDLC applications in Europe in the frame of the EUROCONTROL LINK2000+ Programme. The amended CPDLC mitigates the risk of undetected mis-delivery of messages to the end user and the risk of delivery of undetected corrupt messages to the end user. In addition, the proposed solution reduces the integrity and availability requirements on information provided by the Context Management (CM) application and distributed across different systems during flight.

1.2 The applicability of the amended CPDLC is worldwide as any implementation of CPDLC applications in any region will have similar safety requirements. Therefore, it is envisaged that all CPDLC implementations will use the amended CDPLC application.

1.3 ACP/WGW at its 1st meeting in June 2005, approved the amendment of the ICAO documents (SARPs and Manuals) with respect to the introduction of the amended CPDLC applications (ACP/WGW1 Meeting Report, Agenda Item 2, Appendices B, C, D, and G).

1.4 Following the decisions in ACP/WGW meeting, ICAO has issued a State Letter (ref AN 7/1.3.89-06/8) covering the introduction of the PM-CPDLC Application material in Annex 10. Furthermore, section 2.5 of the State Letter raises the question to maintain or not the currently available CPDLC application.

2. DISCUSSION

2.1 The only known implementation of the CPDLC application that is currently ongoing or planned is in Europe and it is coordinated by EUROCONTROL in the frame of the implementation of initial CPDLC based datalink applications under the LINK 2000+ Programme. Those applications are ACM (ATC Communications Management), ACL (ATC Clearances), AMC(ATC Microphone Check) and DLIC (Data Link Initiation Capability). All these applications are based on Doc 9705.

2.2 In the Pioneer phase of the LINK2000+ datalink applications implementation, previous editions of Doc 9705 were used as the baseline reference. However, in the fully operational phase of the programme, it is the amended CPDLC version which will be implemented and the previous Pioneer implementation of CPDLC will be migrated to the amended application.

2.3 During the transition period both CPDLC versions will be supported by the ground systems. The support for the current CPDLC will cease as soon as the migration is complete. The two versions of the CPDLC applications use distinct identifiers (App-Type 02 for the current CPDLC and App-Type 22 for the amended CPDLC) and therefore the ground systems are able to communicate appropriately with the equipped aircraft.

2.4 In relation to the State Letter question about to maintain or not the current version of the CPDLC application, EUROCONTROL has discussed this with all the relevant Stakeholders of the LINK2000+ (ANSPs, airlines and, industry). It is the agreed view that only the amended CPDLC application meets the safety requirements of a fully operational application and only the amended CPDLC application should be implemented in any large scale operational implementation. Furthermore, maintaining two CPDLC application descriptions in Doc 9705 is redundant and potentially confusing, and in addition it may lead to potential interoperability issues if the two types of CPDLC applications were both operationally implemented.

2.5 Therefore it is proposed to delete the current CPDLC application description and replace it with the amended CPDLC application description as presented in ACP/WGW.

2.6 The proposal to replace the existing CPDLC application description with the amended one has been co-ordinated with the chairman of ACP/WGN, the air-ground applications subgroup of WGN (SWG-N2) and the ATN Maintenance Subgroup (ATNMSG).

3. SUMMARY OF PROPOSED AMENDMENTS TO CPDLC SECTION OF DOC 9705

3.1 Replace Doc 9505, Sub-Volume II, section 2.3 (existing CPDLC, App-Type 2) with the amended CPDLC (previously referred as PM-CPDLC, App-Type 22) appearing as section 2.5 in the approved amendments of ACP/WGW/1 meeting. Renumber paragraphs and cross-references in section 2.5 accordingly.

3.2 Replace the term “PM CPDLC” by “CPDLC” globally in Doc 9705, so that there will only be one, unambiguous CPDLC application, which is the variant currently known as PM-CPDLC. The application type will remain as 22.

3.3 Replace in Sub-Volume IX, Table 9-2, for App-Type 2, the term “CPDLC” with “Reserved.”

3.4 The amended CPDLC material that is proposed to replace the existing section 2.3 of sub volume II of DOC 9705 is provided in Appendix A of this working paper.

4. ACTION BY THE MEETING

4.1 The ACP WGM is invited to approve the proposed amendments to Doc 9705 in relation to the CPDLC application.

ACP-WGW_CPDLC APPENDIX A

2.3 CONTROLLER PILOT DATA LINK COMMUNICATION APPLICATION

2.3.1 INTRODUCTION

2.3.1.1 Overview

2.3.1.1.1 The CPDLC (CPDLC) application allows data link communication between controllers and pilots together with end-to-end protection of message integrity by an application level integrity check that also provides assurance of correct delivery.

2.3.1.1.2 The CPDLC application provides the capability to establish, manage, and terminate CPDLC dialogues between ATS ground and aircraft system peers. Once a dialogue is established, CPDLC provides for controller/pilot message exchange.

2.3.1.1.3 The CPDLC application also provides the capability to establish, manage, and terminate CPDLC dialogues between two ATC ground system peers for the purpose of ground/ground forwarding of a CPDLC Message.

Note 1.— Protection against mis-direction and corruption is not provided on ground forward messages.

Note 2.— Structure:

a) 2.3.1: INTRODUCTION contains the structure of 2.3 and a summary of the functional capabilities of CPDLC.

b) 2.3.2: GENERAL REQUIREMENTS contains the CPDLC version number, and error processing requirements.

c) 2.3.3: ABSTRACT SERVICE DEFINITION contains the description of the abstract service provided by the CPDLC Application Service Element (CPDLC-ASE).

d) 2.3.4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by CPDLC ASEs using Abstract Syntax Notation Number One (ASN.1).

e) 2.3.5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the CPDLC protocol, as well as time constraints and CPDLC-ASE protocol descriptions and state tables.

f) 2.3.6: COMMUNICATION REQUIREMENTS contains the requirements that the CPDLC application imposes on the underlying communication system.

g) 2.3.7: CPDLC USER REQUIREMENTS contains requirements imposed on the user of the CPDLC ASE service and message description tables.

h) 2.3.8: SUBSETTING RULES defines the conformant subsets for the CPDLC-ASE.

Note 3.— Functional Descriptions

*a) 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 :*

1) general information exchange;

2) clearance

i) delivery,

ii) request, and

iii) response;

3) altitude/identity surveillance;

4) monitoring of current/planned position;

5) advisories

i) request and

ii) delivery;

6) system management functions; and

7) emergency situations.

*b) The **Transfer of Data Authority Function** provides the capability for the current data authority to designate another ground system as the next data authority. A CPDLC dialogue can be opened with or by the next data authority at a time before becoming the current data authority. This capability is intended to prevent a loss of communication that would occur if the next data authority were prevented from actually setting up a dialogue with an aircraft until it became the current data authority. The designation of a next data authority is accomplished using a CPDLC message.*

*c) The **Down Stream Clearance Function** provides the capability for an aircraft to contact an air traffic service unit which is not the current data authority for the purpose of receiving a down stream clearance. This information is exchanged using CPDLC message(s).*

*d) The **Ground Forward Function** provides the capability for a ground system to forward information received in a CPDLC message to another ground system. The ground forwarding function can be used by the controlling data authority to forward an aircraft request to the next data authority, so that an aircraft does not need to*

issue the same request again. This function can also be used by a downstream data authority to pass a message to a current data authority for transmission by the current data authority to an aircraft. This information is exchanged using CPDLC message(s). It is a one-way forwarding of information with an indication of success, failure or non-support from the receiving ground system. The CPDLC Ground Forward function does not use the CPDLC Application Message Integrity Check.

Note 4.— See 2.3.7 for detailed CPDLC message intent/use descriptions.

2.3.2 GENERAL REQUIREMENTS

2.3.2.1 CPDLC ASE Version Number

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

2.3.2.2 Error Processing Requirements

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

2.3.2.2.2 In the event of a CPDLC-user invoking a CPDLC service primitive when the CPDLC-ASE is not in a state specified in 2.3.5, the CPDLC-user shall be notified.

2.3.3 THE ABSTRACT SERVICE

2.3.3.1 Service Description

2.3.3.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service shall exhibit external behavior consistent with having implemented a CPDLC-ground-ASE, or CPDLC-air-ASE respectively, with the following abstract service interface primitives, making them available to the CPDLC-ground-user or CPDLC-air-user respectively.

Note 1.— There is no requirement to implement the service in a CPDLC product; however, it is necessary to implement the ground based and air based system in such a way that it will be impossible to detect (from the peer system) whether or not the interface has been built.

Note 2.— This chapter defines the abstract service interface for the CPDLC service. The CPDLC-ASE abstract service is described in this chapter from the viewpoint of the CPDLC-air-user, the CPDLC-ground-user and the CPDLC-service-provider.

Note 3.— This chapter defines the static behavior (i.e., the format) of the CPDLC abstract service. Its dynamic behavior (i.e., how it is used) is described in 2.3.7.

Note 4.— Figure 2.3.3-1 shows the functional model of the CPDLC Application. The functional modules identified in this model are the following :

a) the CPDLC-user,

b) the CPDLC Application Entity (CPDLC-AE) service,

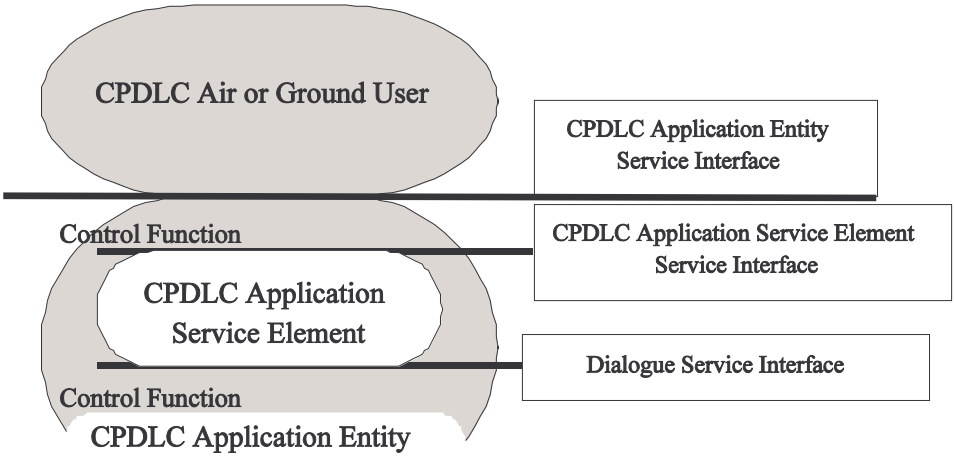
c) the CPDLC-AE,

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

e) the CPDLC Application Service Element (CPDLC-ASE) service,

f) the CPDLC-ASE, and

g) the Dialogue Service (DS).



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 difference with the Standard Mode CPDLC is that 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.

2.3.3.2 The CPDLC-ASE Abstract Service

2.3.3.2.1 The CPDLC-ASE abstract service shall consist of a subset of the following services as allowed in 2.3.8:

a) CPDLC-start service as defined in 2.3.3.3,

b) DSC-start service as defined in 2.3.3.4,

c) CPDLC-message service as defined in 2.3.3.5,

d) CPDLC-end service as defined in 2.3.3.6,

e) DSC-end service as defined in 2.3.3.7,

f) CPDLC-forward service as defined in 2.3.3.8,

g) CPDLC-user-abort service as defined in 2.3.3.9, and

h) CPDLC-provider-abort service as defined in 2.3.3.10.

Note 1.— For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables in 2.3.3.

a)	***blank***	*not present;*
b)	***C***	*conditional upon some predicate explained in the text;*
c)	***C(=)***	*conditional upon the value of the parameter to the left being present, and equal to that value;*
d)	***M***	*mandatory;*
e)	***M(=)***	*mandatory, and equal to the value of the parameter to the left;*
f)	***U***	*user option.*

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

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

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

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

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

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

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

Note 5.— An abstract syntax is a syntactical description of a parameter which does not imply a specific implementation. Only when the CPDLC-ASE maps a parameter into an APDU field, or vice versa, the abstract syntax of the parameter is described by using ASN.1 of 2.3.4 for this field.

2.3.3.3 CPDLC-start Service

Note 1.— The CPDLC-start service is used by the CPDLC-air-user or CPDLC-ground-user to establish a CPDLC dialogue. It is a confirmed service.

Note 2.— Once a CPDLC dialogue is established it remains open until explicitly closed. (See CPDLC-end and CPDLC-abort services.)

2.3.3.3.1 The CPDLC-start service shall contain the primitives and parameters as presented in Table 2.3.3-1.

Table 2.3.3-1. CPDLC-start Service Parameters

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

2.3.3.3.2 Called Peer Identifier

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

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

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

2.3.3.3.2.2 If the service is air initiated, the *Called Peer Identifier* parameter value shall conform to the abstract syntax four to eight-character facility designation.

2.3.3.3.3 Calling Peer Identifier

Note 1.— If the service is ground initiated, this parameter contains the sending ground system's facility designation.

Note 2.— If the service is air initiated, this parameter contains the sending aircraft's 24-bit aircraft address.

2.3.3.3.3.1 If the service is ground initiated, the *Calling Peer Identifier* parameter value shall conform to the abstract syntax four to eight-character facility designation.

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

2.3.3.3.4 Protected CPDLC Message

Note.— The CPDLC-user uses this parameter to send a Protected CPDLC Message to its peer user.

2.3.3.3.4.1 If the CPDLC-start request primitive is invoked by the CPDLC-ground-user, the *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedUplinkMessage.

2.3.3.3.4.2 If the CPDLC-start request primitive is invoked by the CPDLC-air-user, the *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedDownlinkMessage.

Note.— A Protected CPDLC Message may contain no CPDLC Message.

2.3.3.3.5 Response Message

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.

2.3.3.3.5.1 If the CPDLC-start response primitive is invoked by the CPDLC-ground-user, the *Response* Message parameter value shall conform to the ASN.1 abstract syntax ProtectedUplinkMessage.

2.3.3.3.5.2 If the CPDLC-start response primitive is invoked by the CPDLC-air-user, the *Response* Message parameter value shall conform to the ASN.1 abstract syntax ProtectedDownlinkMessage.

Note.— A Response may, and often does, contain no CPDLC Message.

2.3.3.3.6 Result

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

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

2.3.3.3.7 Class of Communication Service

Note 1.— This parameter contains the value of the required class of communication service. If not specified by the CPDLC-user, this indicates that there is no routing preference.

Note 2.— This parameter is used by the CPDLC-ground-user to determine if the Class of Communication value is acceptable for the establishment of a CPDLC dialogue.

Note 3.— The parameter indicated to 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.

2.3.3.3.7.1 Where specified by the CPDLC-user, the *Class of Communication Service* parameter shall have one of the following abstract values: “A”, “B”, “C”, “D”, “E”, “F”, “G”, or “H”.

2.3.3.3.7.2 When this parameter is provided by the 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.

2.3.3.3.8 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.*

2.3.3.3.8.1 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.

2.3.3.4 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.)

2.3.3.4.1 The DSC-start service shall contain the primitives and parameters as presented in Table 2.3.3-2.

Table 2.3.3-2. DSC-start Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Facility Designation	M			
Aircraft Address	M	M(=)		
Protected CPDLC Message	M	M(=)		
Response Message			M	M(=)
Result			M	M(=)

Class of Communication Service	U	M		
Security Required	U			

2.3.3.4.2 Facility Designation

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

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

2.3.3.4.3 Aircraft Address

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

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

2.3.3.4.4 Protected CPDLC Message

Note.— The CPDLC-air-user uses this parameter to send a Protected CPDLC Message to a CPDLC-ground-user.

2.3.3.4.4.1 The *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedDownlinkMessage.

Note.— A Protected CPDLC Message may contain no CPDLC Message.

2.3.3.4.5 Response Message

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.

2.3.3.4.5.1 The *Response* Message parameter shall conform to the ASN.1 abstract syntax ProtectedUplinkMessage.

Note.— A Response may, and often does contain no CPDLC Message.

2.3.3.4.6 Result

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

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

2.3.3.4.7 Class of Communication Service

Note 1.— This parameter contains the value of the required class of communication service. If not specified by the CPDLC-air-user, this indicates that there is no routing preference.

Note 2.— This parameter is used by the CPDLC-ground-user to determine if the Class of Communication value is acceptable for the establishment of a 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.

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

2.3.3.4.7.2 When this parameter is provided by the 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.

2.3.3.4.8 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.*

2.3.3.4.8.1 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.

2.3.3.5 CPDLC-message Service

Note.— The CPDLC-message service can be used for pilot/controller message exchange, once a dialogue is established. It is an unconfirmed service.

2.3.3.5.1 The CPDLC-message service shall contain the primitives and parameters as presented in Table 2.3.3-3.

Table 2.3.3-3. CPDLC-message Service Parameters

Parameter Name	Req	Ind
Protected CPDLC Message	M	M(=)

2.3.3.5.2 Protected CPDLC Message

Note.— This parameter contains a Protected *CPDLC Message.*

2.3.3.5.2.1 If the CPDLC-message service is invoked by the CPDLC-ground-user, the *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedUplinkMessage.

2.3.3.5.2.2 If the CPDLC-message service is invoked by the CPDLC-air-user, the *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedDownlinkMessage.

Note.— This Protected CPDLC Message provided by the CPDLC-user always contains a CPDLC Message.

2.3.3.6 CPDLC-end Service

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

2.3.3.6.1 The CPDLC-end service shall contain the primitives and parameters as presented in Table 2.3.3-4.

Table 2.3.3-4. CPDLC-end Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Protected CPDLC Message	M	M(=)	M	M(=)
Result			M	M(=)

2.3.3.6.2 Protected CPDLC Message

Note.— This parameter contains a Protected *CPDLC Message.*

2.3.3.6.2.1 The *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedUplinkMessage, if provided by the CPDLC-ground-user.

2.3.3.6.2.2 The *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedDownlinkMessage, if provided by the CPDLC-air-user.

Note.— A Protected CPDLC Message may contain no CPDLC Message.

2.3.3.6.3 Result

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

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

2.3.3.7 DSC-end Service

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

2.3.3.7.1 The DSC-end service shall contain the primitives and parameters as presented in Table 2.3.3-5.

Table 2.3.3-5. DSC-end Service Parameters

Parameter Name	**Req**	**Ind**	**Rsp**	**Cnf**
Protected CPDLC Message	M	M(=)	M	M(=)
Result			M	M(=)

2.3.3.7.2 Protected CPDLC Message

Note.— This parameter contains a Protected *CPDLC Message.*

2.3.3.7.2.1 The *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedUplinkMessage, if provided by the CPDLC-ground-user.

2.3.3.7.2.2 The *Protected CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ProtectedDownlinkMessage if provided by the CPDLC-air-user.

Note.— A Protected CPDLC Message may contain no CPDLC Message.

2.3.3.7.3 Result

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

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

2.3.3.8 CPDLC-forward Service

Note.— The CPDLC-forward service is used by a CPDLC-ground-user to send a CPDLC Message to another CPDLC-ground-user. Its primary use is for the forwarding of aircraft requests.

2.3.3.8.1 If the CPDLC-forward service is supported by the receiving ground system and the sending CPDLC-ground-ASE and receiving CPDLC-ground-ASE version numbers are equal, the CPDLC-forward service shall contain the primitives and parameters as presented in Table 2.3.3-6.

Table 2.3.3-6. CPDLC-forward Service Parameters (Service Supported, Versions Equal)

Parameter Name	Req	Ind	Cnf
Called Facility Designation	M		
Calling Facility Designation	M	M(=)	
CPDLC Message	M	M(=)	
Class of Communication Service	U		
Security Required	U		
Result			M

2.3.3.8.2 If the CPDLC-forward service is not supported by the receiving ground system, or if the CPDLC-forward service is supported by the receiving ground system but the sending CPDLC-ground-ASE and receiving CPDLC-ground-ASE version numbers are not equal, the CPDLC-forward service shall contain the primitives and parameters as presented in Table 2.3.3-7.

Table 2.3.3-7. CPDLC-forward Service Parameters (Service Not Supported or Versions Not Equal)

Parameter Name	Req	Cnf
Called Facility Designation	M	
Calling Facility Designation	M	
ASE Version Number		C
CPDLC Message	M	
Class of Communication Service	U	
Security Required	U	
Result		M

2.3.3.8.3 Called Facility Designation

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

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

2.3.3.8.4 Calling Facility Designation

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

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

2.3.3.8.5 ASE Version Number

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

2.3.3.8.5.1 When provided by the CPDLC-ground-ASE, the *ASE Version Number* parameter shall conform to the abstract integer value in the range 1-255.

2.3.3.8.5.2 Only if the sending CPDLC-ground-ASE version number is not equal to the receiving CPDLC-ground-ASE version number shall the receiving CPDLC-ground-ASE version number be confirmed to the sending CPDLC-ground-user.

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

2.3.3.8.6 CPDLC Message

Note.— The sending CPDLC-ground-user uses this parameter to forward a CPDLC Message to another CPDLC-ground-user.

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

2.3.3.8.7 Class of Communication Service

Note.— This parameter contains the value of the required class of communication service. If not specified by the CPDLC-ground-user, this indicates that there is no routing preference.

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

2.3.3.8.8 Result

Note.— This parameter contains the result of the CPDLC-forward service. It will indicate success (service supported and matching versions), service unsupported, or version number incompatibility.

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

2.3.3.8.9 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.*

2.3.3.8.9.1 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.

2.3.3.9 CPDLC-user-abort Service

Note 1.— This service provides the capability for either the CPDLC-air-user or a CPDLC-ground-user to abort communication with its peer. It can be invoked at any time the 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.

2.3.3.9.1 The CPDLC-user-abort service shall contain the primitives and parameters as presented in Table 2.3.3-8.

Table 2.3.3-8. CPDLC-user-abort Service Parameters

Parameter Name	Req	Ind
Reason	U	M

2.3.3.9.2 Reason

Note 1.— This parameter is used to indicate a reason for aborting the CPDLC or DSC dialogue.

Note 2.— If provided by the CPDLC-user, the parameter indicated to the peer CPDLC-user is that provided by the CPDLC-user, else it is what the ASE supplies.

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

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

2.3.3.10 CPDLC-provider-abort Service

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

2.3.3.10.1 The CPDLC-provider-abort service shall contain the primitives and parameters as presented in Table 2.3.3-9.

Table 2.3.3-9. CPDLC-provider-abort Service Parameters

Parameter Name	**Ind**
Reason	M

2.3.3.10.2 Reason

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

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

2.3.4 FORMAL DEFINITIONS OF MESSAGES

2.3.4.1 Encoding/Decoding Rules

2.3.4.1.1 A CPDLC-air-ASE shall be capable of encoding AircraftPDUs APDUs and decoding GroundPDUs APDUs as defined in the ASN.1 module CPDLCAPDUsVersion1 specified in 2.3.4.2.

Note.— The ProtectedUplinkMessage and ProtectedDownlinkMessage ASN.1 types both contain 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-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.

2.3.4.1.2 The CPDLC-air-user shall be capable of encoding ATCDownlinkMessage and decoding ATCUplinkMessage types, as defined in the ASN.1 module CPDLCMessageSetVersion1 specified in 2.3.4.3.

2.3.4.1.3 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 2.3.4.2.

Note 1.— The ProtectedUplinkMessage and ProtectedDownlinkMessage ASN.1 types both contain 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.

2.3.4.1.4 The CPDLC-ground-user shall be capable of encoding ATCUplinkMessage and decoding ATCDownlinkMessage types, as defined in the ASN.1 module CPDLCMessageSetVersion1 specified in 2.3.4.3.

2.3.4.1.5 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 2.3.4.3.

2.3.4.2 CPDLC Message ASN.1 Abstract Syntax

2.3.4.2.1 The abstract syntax of the CPDLC protocol data units shall comply with the description contained in the ASN.1 module 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]     ProtectedUplinkMessage,
        send                    [3]     ProtectedUplinkMessage,
        forward                 [4]     ATCForwardMessage,
        forwardresponse         [5]     ATCForwardResponse,
        ...
        }

ProtectedUplinkMessage::= SEQUENCE
        {
        algorithmIdentifier     [0] AlgorithmIdentifier OPTIONAL,
        protectedMessage        [1] CPDLCMessage 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]     ProtectedDownlinkMessage,
        ...
        }

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

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


ProtectedDownlinkMessage::= SEQUENCE
        {
        algorithmIdentifier             [0] AlgorithmIdentifier OPTIONAL,
        protectedMessage                [1] CPDLCMessage 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)}

CPDLCMessage ::= 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
```

2.3.4.3 CPDLC Message ASN.1 Abstract Syntax

2.3.4.3.1 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] uM106Speed	Urg(N)/Alr(M)/Resp(W/U) [106] Speed,
--	MAINTAIN PRESENT SPEED uM107NULL	Urg(N)/Alr(M)/Resp(W/U) [107] NULL,
--	MAINTAIN [speed] OR GREATER uM108Speed	Urg(N)/Alr(M)/Resp(W/U) [108] Speed,
--	MAINTAIN [speed] OR LESS uM109Speed	Urg(N)/Alr(M)/Resp(W/U) [109] Speed,
--	MAINTAIN [speed] TO [speed] uM110SpeedSpeed	Urg(N)/Alr(M)/Resp(W/U) [110] SpeedSpeed,
--	INCREASE SPEED TO [speed] uM111Speed	Urg(N)/Alr(M)/Resp(W/U) [111] Speed,
--	INCREASE SPEED TO [speed] OR GREATER uM112Speed	Urg(N)/Alr(M)/Resp(W/U) [112] Speed,
--	REDUCE SPEED TO [speed] uM113Speed	Urg(N)/Alr(M)/Resp(W/U) [113] Speed,
--	REDUCE SPEED TO [speed] OR LESS uM114Speed	Urg(N)/Alr(M)/Resp(W/U) [114] Speed,
--	DO NOT EXCEED [speed] uM115Speed	Urg(N)/Alr(M)/Resp(W/U) [115] Speed,
--	RESUME NORMAL SPEED uM116NULL	Urg(N)/Alr(M)/Resp(W/U) [116] NULL,
--	CONTACT [unitname] [frequency] uM117UnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [117] UnitNameFrequency,
--	AT [position] CONTACT [unitname] [frequency] uM118PositionUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [118] PositionUnitNameFrequency,
--	AT [time] CONTACT [unitname] [frequency] uM119TimeUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [119] TimeUnitNameFrequency,
--	MONITOR [unitname] [frequency] uM120UnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [120] UnitNameFrequency,
--	AT [position] MONITOR [unitname] [frequency] uM121PositionUnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) [121] PositionUnitNameFrequency,

```
--      AT [time] MONITOR [unitname] [frequency]        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] uM185PositionLevel	Urg(N)/Alr(M)/Resp(W/U) [185] PositionLevel,
--	AFTER PASSING [position] DESCEND TO [level] uM186PositionLevel	Urg(N)/Alr(M)/Resp(W/U) [186] PositionLevel,
--	[freetext] uM187FreeText	Urg(L)/Alr(N)/Resp(N) [187] FreeText,
--	AFTER PASSING [position] MAINTAIN [speed] uM188PositionSpeed	Urg(N)/Alr(M)/Resp(W/U) [188] PositionSpeed,
--	ADJUST SPEED TO [speed] uM189Speed	Urg(N)/Alr(M)/Resp(W/U) [189] Speed,
--	FLY HEADING [degrees] uM190Degrees	Urg(N)/Alr(M)/Resp(W/U) [190] Degrees,
--	ALL ATS TERMINATED uM191NULL	Urg(N)/Alr(M)/Resp(R) [191] NULL,
--	REACH [level] BY [time] uM192LevelTime	Urg(N)/Alr(M)/Resp(W/U) [192] LevelTime,
--	IDENTIFICATION LOST uM193NULL	Urg(N)/Alr(M)/Resp(R) [193] NULL,
--	[freetext] uM194FreeText	Urg(N)/Alr(L)/Resp(Y) [194] FreeText,
--	[freetext] uM195FreeText	Urg(L)/Alr(L)/Resp(R) [195] FreeText,
--	[freetext] uM196FreeText	Urg(N)/Alr(M)/Resp(W/U) [196] FreeText,
--	[freetext] uM197FreeText	Urg(U)/Alr(M)/Resp(W/U) [197] FreeText,
--	[freetext] uM198FreeText	Urg(D)/Alr(H)/Resp(W/U) [198] FreeText,
--	[freetext] uM199FreeText	Urg(N)/Alr(L)/Resp(N) [199] FreeText,
--	REPORT REACHING uM200NULL	Urg(N)/Alr(L)/Resp(W/U) [200] NULL,

--	Not Used uM201NULL	Urg(L)/Alr(L)/Resp(N) [201] NULL,
--	Not Used uM202NULL	Urg(L)/Alr(L)/Resp(N) [202] NULL,
--	[freetext] uM203FreeText	Urg(N)/Alr(M)/Resp(R) [203] FreeText,
--	[freetext] uM204FreeText	Urg(N)/Alr(M)/Resp(Y) [204] FreeText,
--	[freetext] uM205FreeText	Urg(N)/Alr(M)/Resp(A/N) [205] FreeText,
--	[freetext] uM206FreeText	Urg(L)/Alr(N)/Resp(Y) [206] FreeText,
--	[freetext] uM207FreeText	Urg(L)/Alr(L)/Resp(Y) [207] FreeText,
--	[freetext] uM208FreeText	Urg(L)/Alr(L)/Resp(N) [208] FreeText,
--	REACH [level] BY [position] uM209LevelPosition	Urg(N)/Alr(M)/Resp(W/U) [209] LevelPosition,
--	IDENTIFIED [position] uM210Position	Urg(N)/Alr(M)/Resp(R) [210] Position,
--	REQUEST FORWARDED uM211NULL	Urg(N)/Alr(L)/Resp(N) [211] NULL,
--	[facilitydesignation] ATIS [atiscode] CURRENT uM212FacilityDesignationATISCode	Urg(N)/Alr(L)/Resp(R) [212] FacilityDesignationATISCode,
--	[facilitydesignation] ALTIMETER [altimeter] uM213FacilityDesignationAltimeter	Urg(N)/Alr(L)/Resp(R) [213] FacilityDesignationAltimeter,
--	RVR RUNWAY [runway] [rvr] uM214RunwayRVR	Urg(N)/Alr(M)/Resp(R) [214] RunwayRVR,
--	TURN [direction][degrees] uM215DirectionDegrees	Urg(N)/Alr(M)/Resp(W/U) [215] DirectionDegrees,
--	REQUEST FLIGHT PLAN uM216NULL	Urg(N)/Alr(M)/Resp(Y) [216] NULL,

--	REPORT ARRIVAL uM217NULL	Urg(N)/Alr(M)/Resp(Y) [217] NULL,
--	REQUEST ALREADY RECEIVED uM218NULL	Urg(L)/Alr(N)/Resp(N) [218] NULL,
--	STOP CLIMB AT [level] uM219Level	Urg(U)/Alr(M)/Resp(W/U) [219] Level,
--	STOP DESCENT AT [level] uM220Level	Urg(U)/Alr(M)/Resp(W/U) [220] Level,
--	STOP TURN HEADING [degrees] uM221Degrees	Urg(U)/Alr(M)/Resp(W/U) [221] Degrees,
--	NO SPEED RESTRICTION uM222NULL	Urg(L)/Alr(L)/Resp(R) [222] NULL,
--	REDUCE TO MINIMUM APPROACH SPEED uM223NULL	Urg(N)/Alr(M)/Resp(W/U) [223] NULL,
--	NO DELAY EXPECTED uM224NULL	Urg(N)/Alr(L)/Resp(R) [224] NULL,
--	DELAY NOT DETERMINED uM225NULL	Urg(N)/Alr(L)/Resp(R) [225] NULL,
--	EXPECTED APPROACH TIME [time] uM226Time	Urg(N)/Alr(L)/Resp(R) [226] Time,
--	LOGICAL ACKNOWLEDGMENT uM227NULL	Urg(N)/Alr(M)/Resp(N) [227] NULL,
--	REPORT ETA [position] uM228Position	Urg(L)/Alr(L)/Resp(Y) [228] Position,
--	REPORT ALTERNATE AERODROME uM229NULL	Urg(L)/Alr(L)/Resp(Y) [229] NULL,
--	IMMEDIATELY uM230NULL	Urg(D)/Alr(H)/Resp(N) [230] NULL,
--	STATE PREFERRED LEVEL uM231NULL	Urg(L)/Alr(L)/Resp(Y) [231] NULL,
--	STATE TOP OF DESCENT uM232NULL	Urg(L)/Alr(L)/Resp(Y) [232] NULL,

```
--      USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED
--                                                      Urg(N)/Alr(M)/Resp(N)
        uM233NULL                                       [233] NULL,

--      FLIGHT PLAN NOT HELD                            Urg(L)/Alr(L)/Resp(N)
        uM234NULL                                       [234] NULL,

--      ROGER 7500                                      Urg(U)/Alr(H)/Resp(N)
        uM235NULL                                       [235] NULL,

--      LEAVE CONTROLLED AIRSPACE                       Urg(N)/Alr(M)/Resp(W/U)
        uM236NULL                                       [236] NULL,

        ...,

--      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] dM37Level	Urg(N)/Alr(L)/Resp(N) [37] Level,
--	ASSIGNED LEVEL [level] dM38Level	Urg(N)/Alr(M)/Resp(N) [38] Level,
--	ASSIGNED SPEED [speed] dM39Speed	Urg(N)/Alr(M)/Resp(N) [39] Speed,
--	ASSIGNED ROUTE [routeClearance] dM40RouteClearance	Urg(N)/Alr(M)/Resp(N) [40] RouteClearanceIndex,
--	BACK ON ROUTE dM41NULL	Urg(N)/Alr(M)/Resp(N) [41] NULL,
--	NEXT WAYPOINT [position] dM42Position	Urg(N)/Alr(L)/Resp(N) [42] Position,
--	NEXT WAYPOINT ETA [time] dM43Time	Urg(N)/Alr(L)/Resp(N) [43] Time,
--	ENSUING WAYPOINT [position] dM44Position	Urg(N)/Alr(L)/Resp(N) [44] Position,
--	REPORTED WAYPOINT [position] dM45Position	Urg(N)/Alr(L)/Resp(N) [45] Position,
--	REPORTED WAYPOINT [time] dM46Time	Urg(N)/Alr(L)/Resp(N) [46] Time,
--	SQUAWKING [code] dM47Code	Urg(N)/Alr(L)/Resp(N) [47] Code,
--	POSITION REPORT [positionreport] dM48PositionReport	Urg(N)/Alr(M)/Resp(N) [48] PositionReport,
--	WHEN CAN WE EXPECT [speed] dM49Speed	Urg(L)/Alr(L)/Resp(Y) [49] Speed,
--	WHEN CAN WE EXPECT [speed] TO [speed] dM50SpeedSpeed	Urg(L)/Alr(L)/Resp(Y) [50] SpeedSpeed,
--	WHEN CAN WE EXPECT BACK ON ROUTE dM51NULL	Urg(L)/Alr(L)/Resp(Y) [51] NULL,
--	WHEN CAN WE EXPECT LOWER LEVEL dM52NULL	Urg(L)/Alr(L)/Resp(Y) [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 ACKNOWLEDGMENT                          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]SpeedTypeSpeedTypeSpeedType
Speed,
        ...
        }
```

AircraftAddress ::= BIT STRING (SIZE(24))

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

Airport ::= IA5String (SIZE (4))

```
Altimeter ::=  CHOICE
        {
        altimeterEnglish        [0]     AltimeterEnglish,
        altimeterMetric         [1]     AltimeterMetric
        }
```

AltimeterEnglish ::= INTEGER (2200..3200)
-- unit = Inches Mercury, Range (22.00 .. 32.00), resolution = 0.01

AltimeterMetric ::= INTEGER (7500..12500)
-- unit = Hectopascal, Range (750.0..1250.0), resolution = 0.1

ATISCode ::= IA5String (SIZE (1))

ATSRouteDesignator ::= IA5String (SIZE (2..7))

```
ATWAlongTrackWaypoint ::= SEQUENCE
        {
        position                [0]     Position,
        aTWDistance             [1]     ATWDistance,
        speed                   [2]     Speed                   OPTIONAL,
        aTWLevels               [3]     ATWLevelSequence        OPTIONAL
        }
```

```
ATWLevel ::= SEQUENCE
        {
        atw             ATWLevelTolerance,
        level           Level
        }
```

ATWLevelSequence ::= SEQUENCE SIZE (1..2) OF ATWLevel

```
ATWLevelTolerance ::= ENUMERATED
        {
        at                  (0),
        atorabove (1),
        atorbelow (2)
        }

ATWDistance ::= SEQUENCE
        {
        atwDistanceTolerance            ATWDistanceTolerance,
        distance                        Distance
        }

ATWDistanceTolerance ::= ENUMERATED
        {
        plus                (0),
        minus               (1)
        }

ClearanceType ::= ENUMERATED
        {
        noneSpecified       (0),
        approach            (1),
        departure           (2),
        further             (3),
        start-up            (4),
        pushback            (5),
        taxi                (6),
        take-off            (7),
        landing             (8),
        oceanic             (9),
        en-route            (10),
        downstream          (11),
        ...
        }

Code ::=  SEQUENCE SIZE (4) OF CodeOctalDigit

CodeOctalDigit ::= INTEGER (0..7)

ControlledTime ::= SEQUENCE
        {
        time                Time,
        timeTolerance       TimeTolerance
        }
```

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

DateTimeGroup ::= SEQUENCE
        {
        date                Date,
        timehhmmss          Timehhmmss
        }

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

DegreeIncrement ::= INTEGER (1..20)
        --unit = Degree, Range (1..20), resolution = 1

Degrees ::=  CHOICE
        {
        degreesMagnetic             [0]     DegreesMagnetic,
        degreesTrue                 [1]     DegreesTrue
        }

DegreesMagnetic ::= INTEGER (1..360)
        --unit = degree, Range (1..360), resolution = 1

DegreesTrue ::= INTEGER (1..360)
        --unit = degree, Range (1..360), resolution = 1

DepartureClearance ::= SEQUENCE
        {
        aircraftFlightIdentification[0]     AircraftFlightIdentification,
        clearanceLimit                      [1]     Position,
        flightInformation                   [2]     FlightInformation       OPTIONAL,
        furtherInstructions                 [3]     FurtherInstructions     OPTIONAL
        }

DepartureMinimumInterval ::= INTEGER (1..150)
        --unit = Minute, Range (0.1..15.0), resolution = 0.1
```

```
Direction ::= ENUMERATED
        {
        left                    (0),
        right                   (1),
        eitherSide      (2),
        north                   (3),
        south                   (4),
        east                    (5),
        west                    (6),
        northEast               (7),
        northWest               (8),
        southEast       (9),
        southWest               (10)
        }

DirectionDegrees ::=  SEQUENCE
        {
        direction       Direction,
        degrees         Degrees
        }

Distance ::= CHOICE
        {
        distanceNm      [0]     DistanceNm,
        distanceKm      [1]     DistanceKm
        }

DistanceKm ::= INTEGER (0..8000)
--      unit = Kilometer, Range (0..2000), resolution = 0.25

DistanceNm ::= INTEGER (0..9999)
--      unit = Nautical Mile, Range (0..999.9), resolution = 0.1

DistanceSpecified ::= CHOICE
        {
        distanceSpecifiedNm     [0]     DistanceSpecifiedNm,
        distanceSpecifiedKm     [1]     DistanceSpecifiedKm
        }

DistanceSpecifiedDirection ::=  SEQUENCE
        {
        distanceSpecified       DistanceSpecified,
        direction               Direction
        }
```

```
DistanceSpecifiedDirectionTime ::=  SEQUENCE
        {
        distanceSpecifiedDirection          DistanceSpecifiedDirection,
        time                                Time
        }

DistanceSpecifiedKm ::= INTEGER (1..500)
--      unit = Kilometer, Range (1..500), resolution = 1

DistanceSpecifiedNm ::= INTEGER (1..250)
--      unit = Nautical Mile, Range (1..250), resolution = 1

ErrorInformation ::=  ENUMERATED
        {
        unrecognizedMsgReferenceNumber                  (0),
        logicalAcknowledgmentNotAccepted                (1),
        insufficientResources                           (2),
        invalidMessageElementCombination                (3),
        invalidMessageElement                           (4),
        ...
        }

Facility ::= CHOICE
        {
        noFacility          [0]     NULL,
        facilityDesignation         [1]     FacilityDesignation
        }

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

FacilityFunction ::= ENUMERATED
        {
        center                      (0),
        approach            (1),
        tower                       (2),
        final                       (3),
        groundControl               (4),
        clearanceDelivery (5),
        departure                   (6),
        control                     (7),
        radio                       (8),
        ...
        }
```

```
FacilityDesignationAltimeter ::= SEQUENCE
        {
        facilityDesignation             FacilityDesignation,
        altimeter                       Altimeter
        }

FacilityDesignationATISCode ::= SEQUENCE
        {
        facilityDesignation             FacilityDesignation,
        aTISCode                ATISCode
        }

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
```

2.3.5 PROTOCOL DEFINITION

2.3.5.1 Sequence Rules

2.3.5.1.1 With the exception of abort primitives, only the sequence of primitives illustrated in figures 2.3.5-1 to 2.3.5-18 shall be permitted.

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

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

Note 3.— Primitives are processed in the order received.

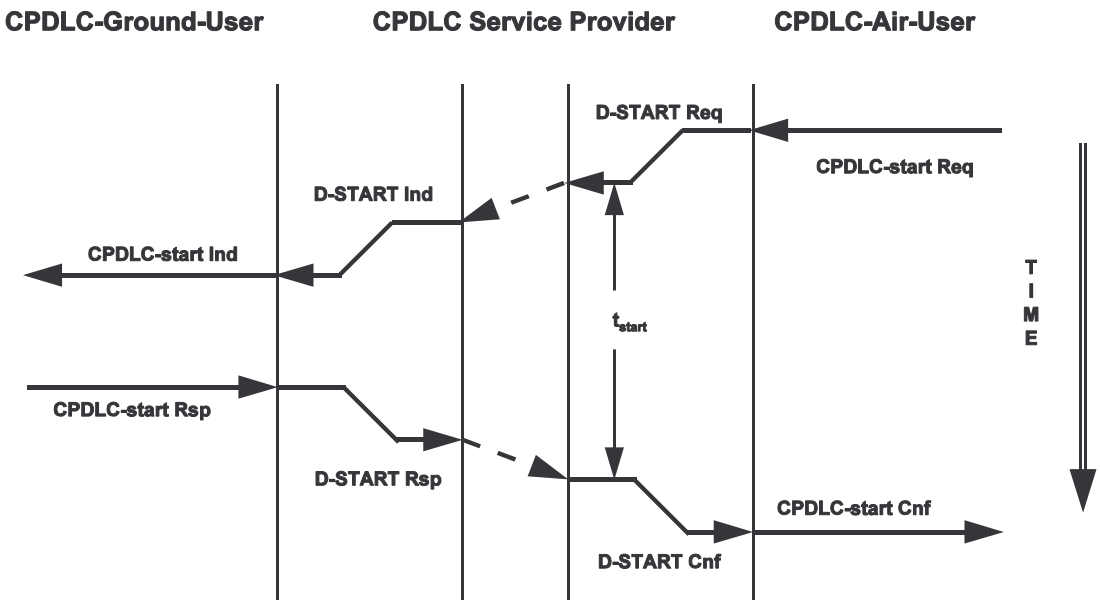


Figure 2.3.5-1. Sequence Diagram for CPDLC-start Service/Air Initiated

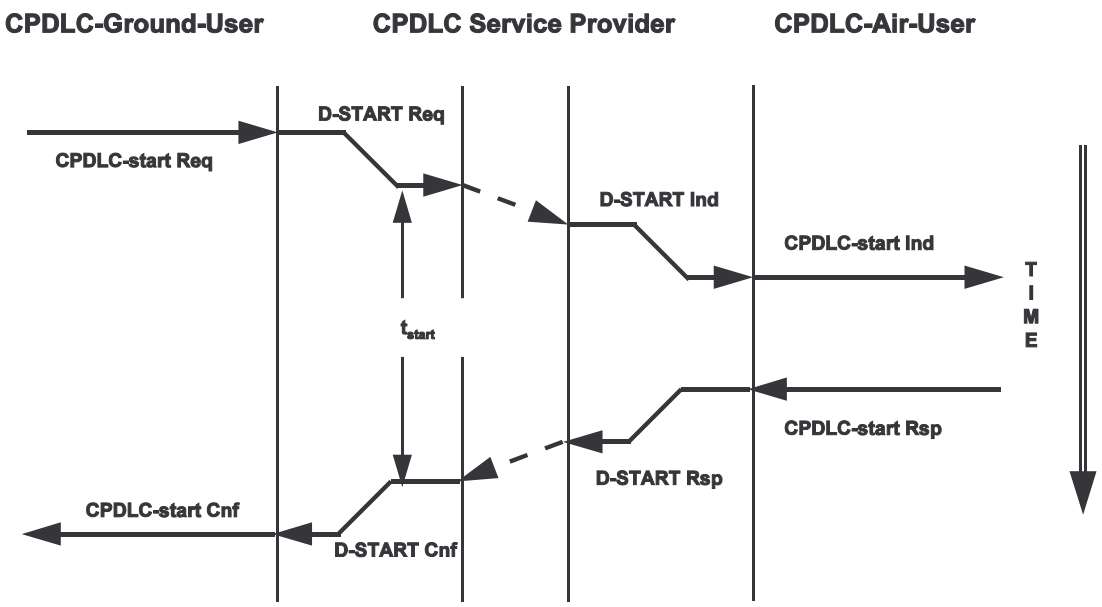


Figure 2.3.5-2. Sequence Diagram for CPDLC-start Service/Ground Initiated

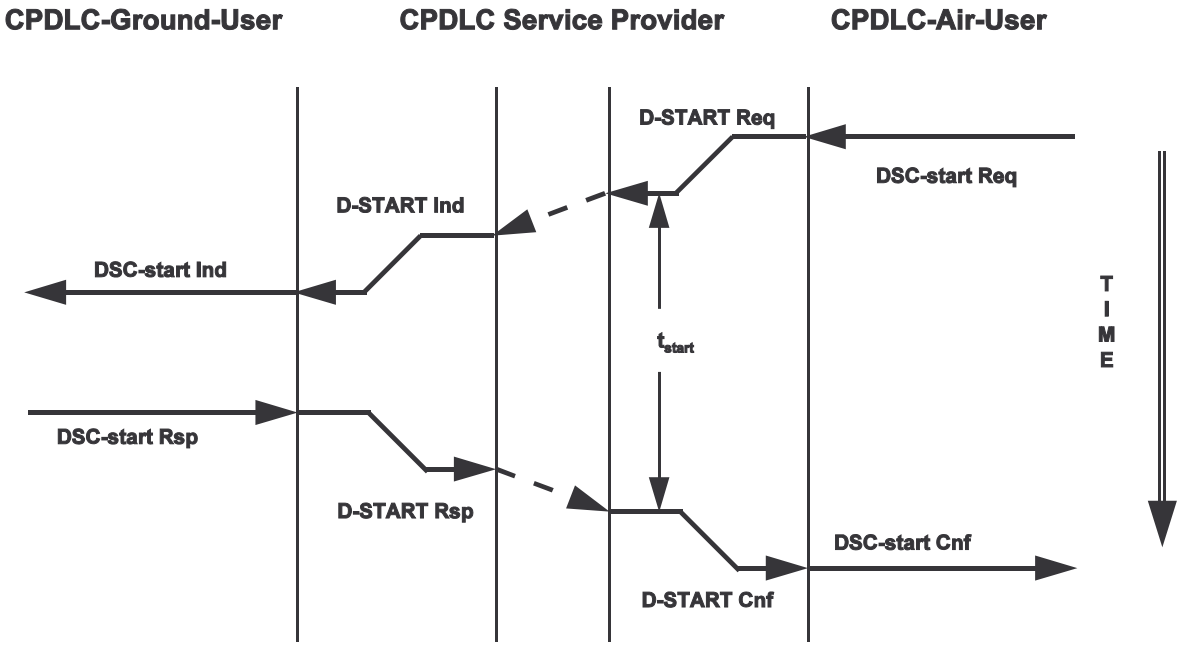


Figure 2.3.5-3. Sequence Diagram for DSC-start Service

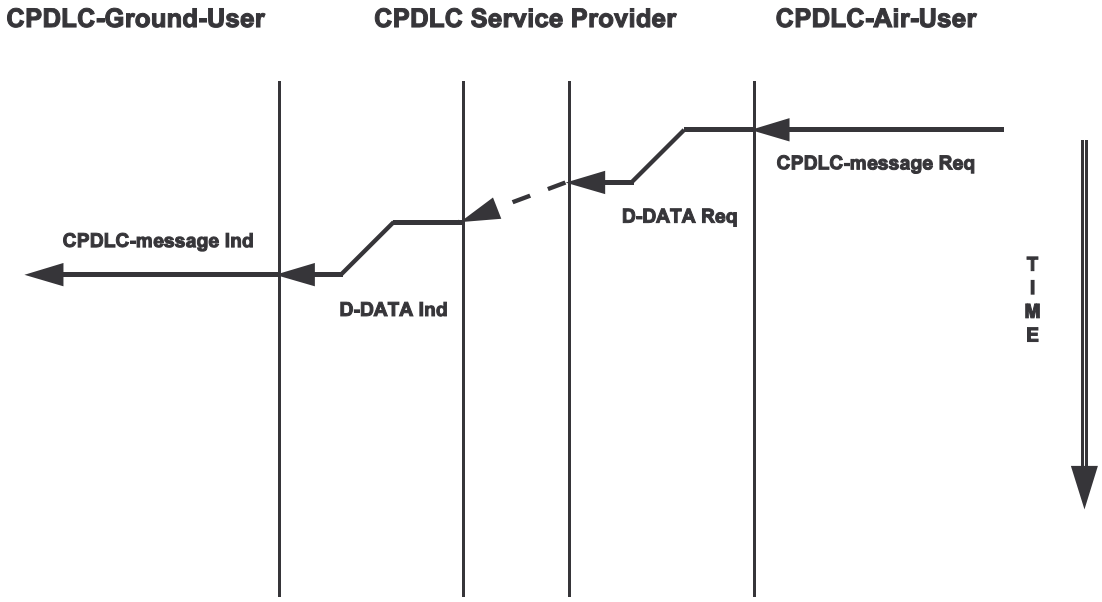


Figure 2.3.5-4. Sequence Diagram for CPDLC-message Service/Air Initiated

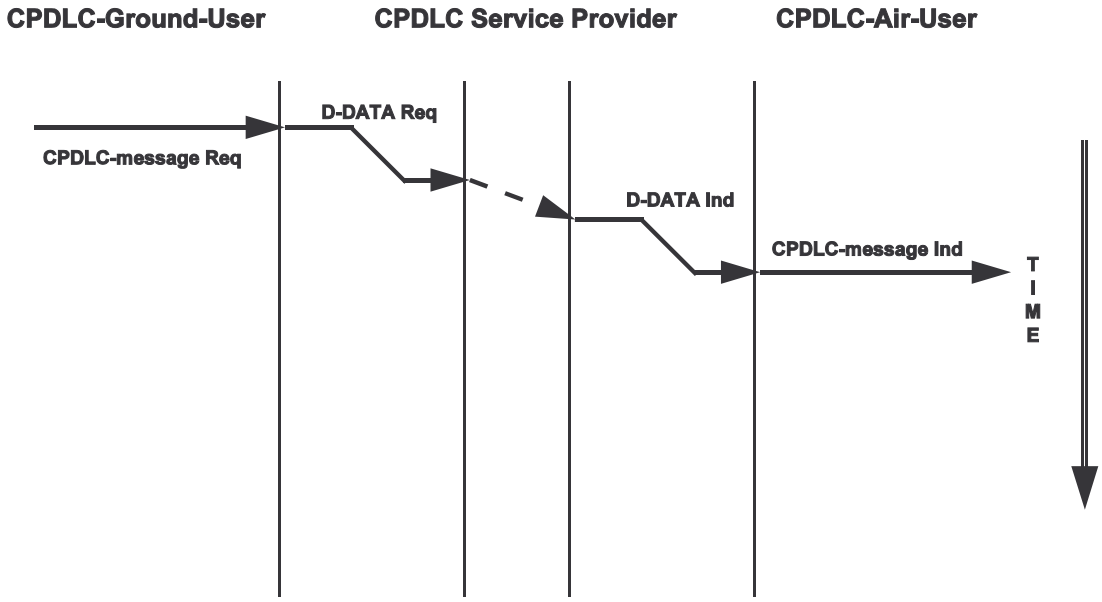


Figure 2.3.5-5. Sequence Diagram for CPDLC-message Service/Ground Initiated

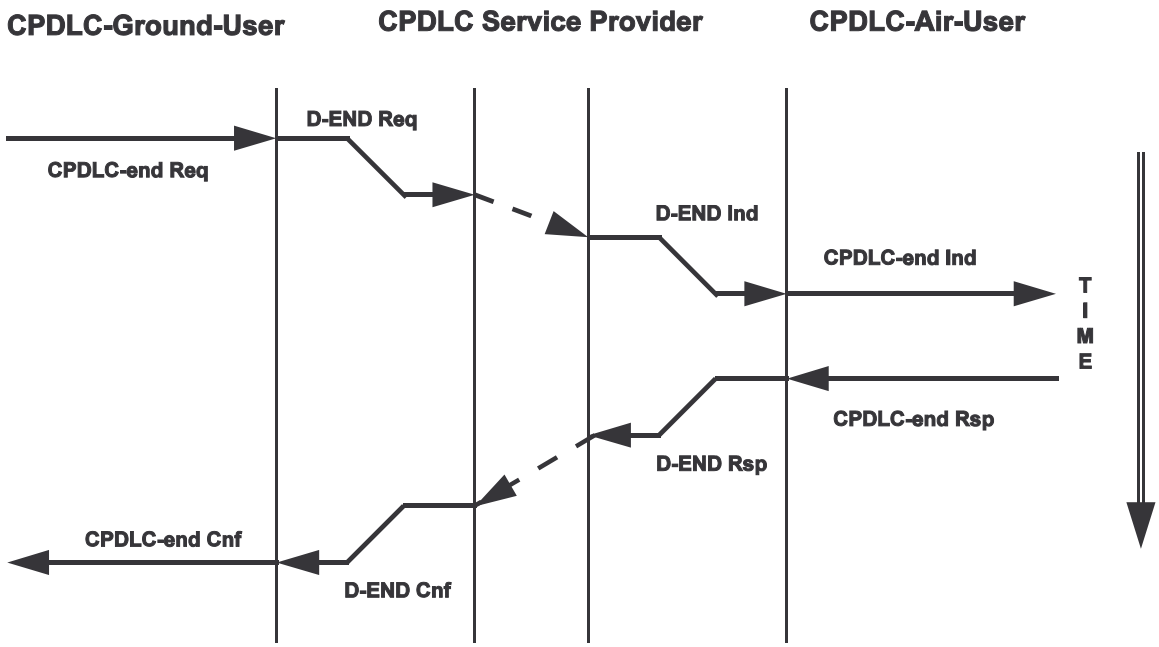


Figure 2.3.5-6. Sequence Diagram for CPDLC-end Service

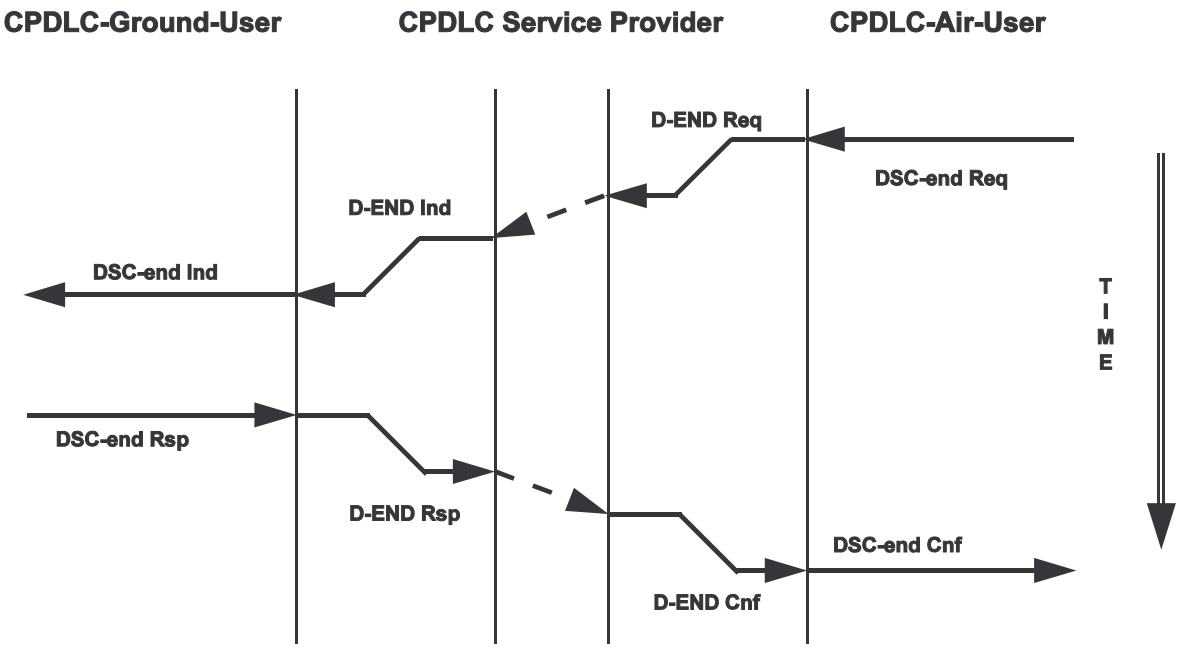


Figure 2.3.5-7. Sequence Diagram for DSC-end Service

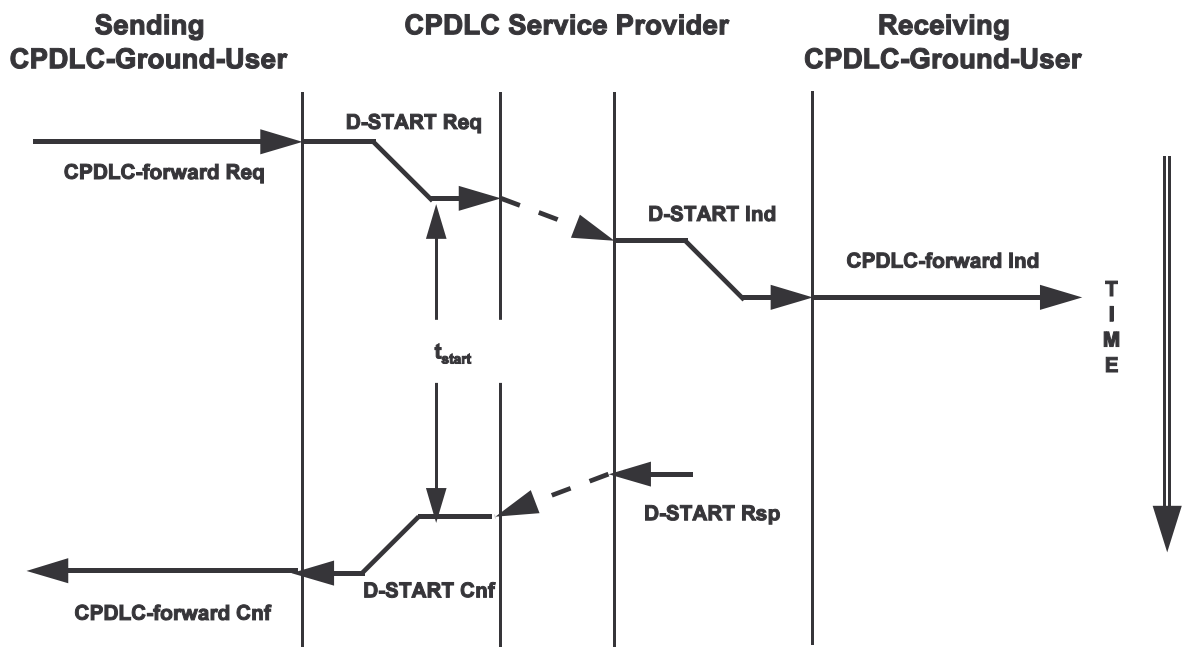


Figure 2.3.5-8. Sequence Diagram for CPDLC-forward Service/Ground Forwarding Supported, ASE Version Numbers the Same

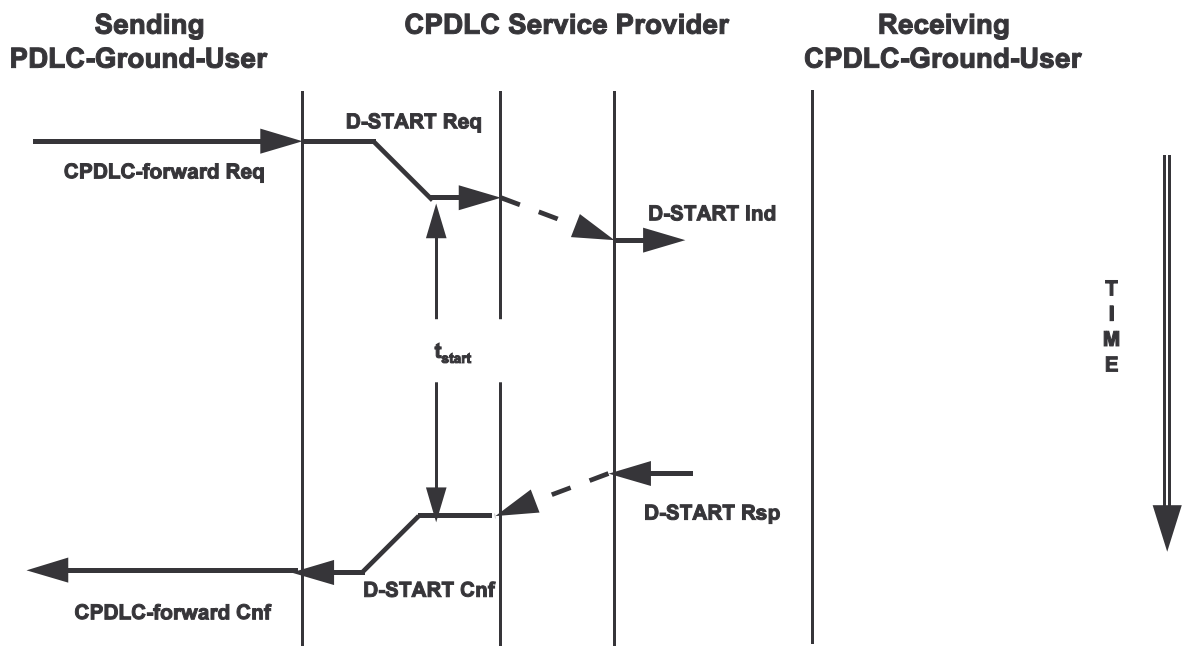


Figure 2.3.5-9. Sequence Diagram for CPDLC-forward Service/Ground Forwarding Not Supported, or Ground Forwarding Supported and ASE Version Numbers Not the Same

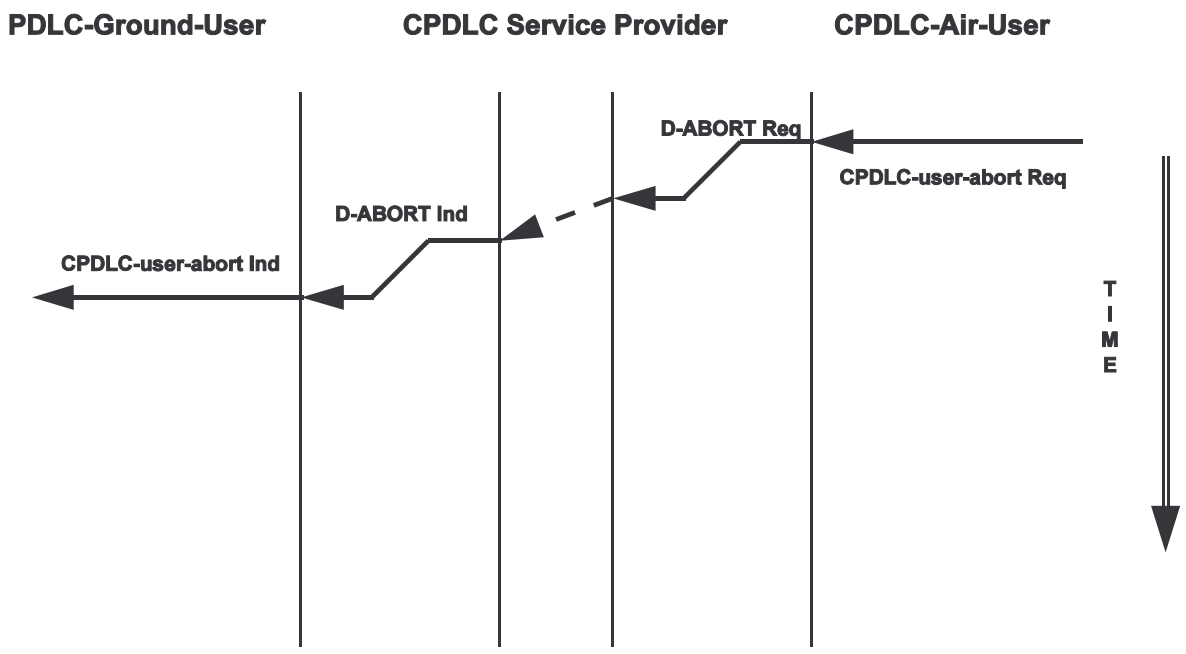


Figure 2.3.5-10. Sequence Diagram for CPDLC-user-abort Service/CPDLC-Air-User Initiated

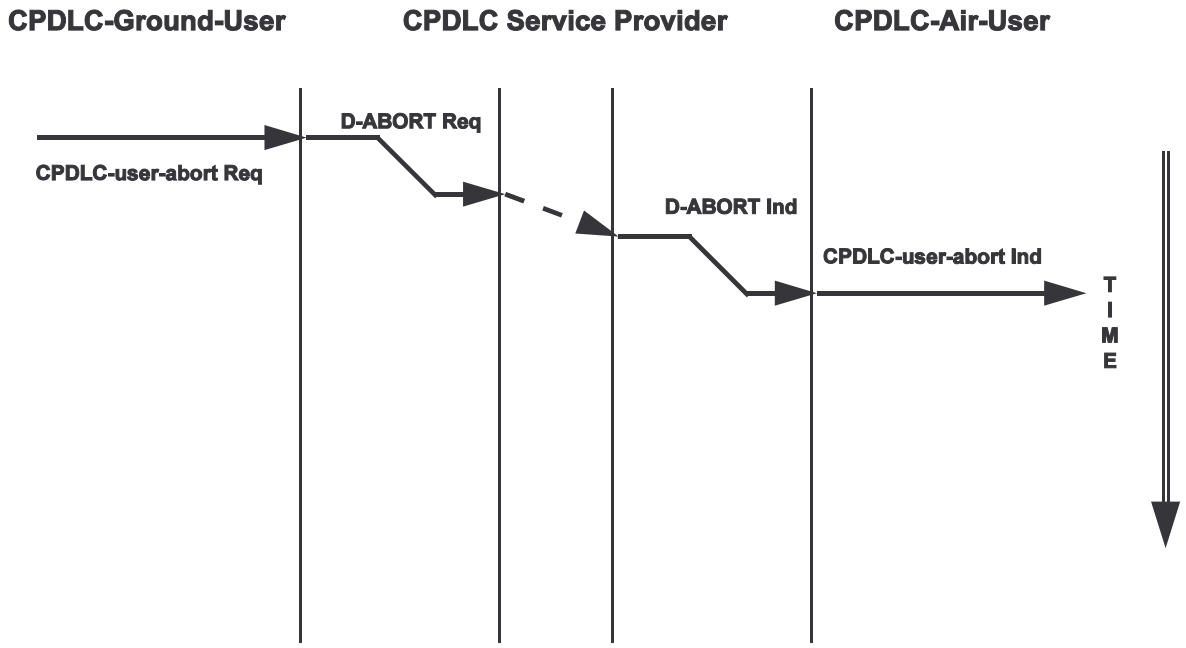


Figure 2.3.5-11. Sequence Diagram for CPDLC-user-abort Service/CPDLC-Ground-User Initiated

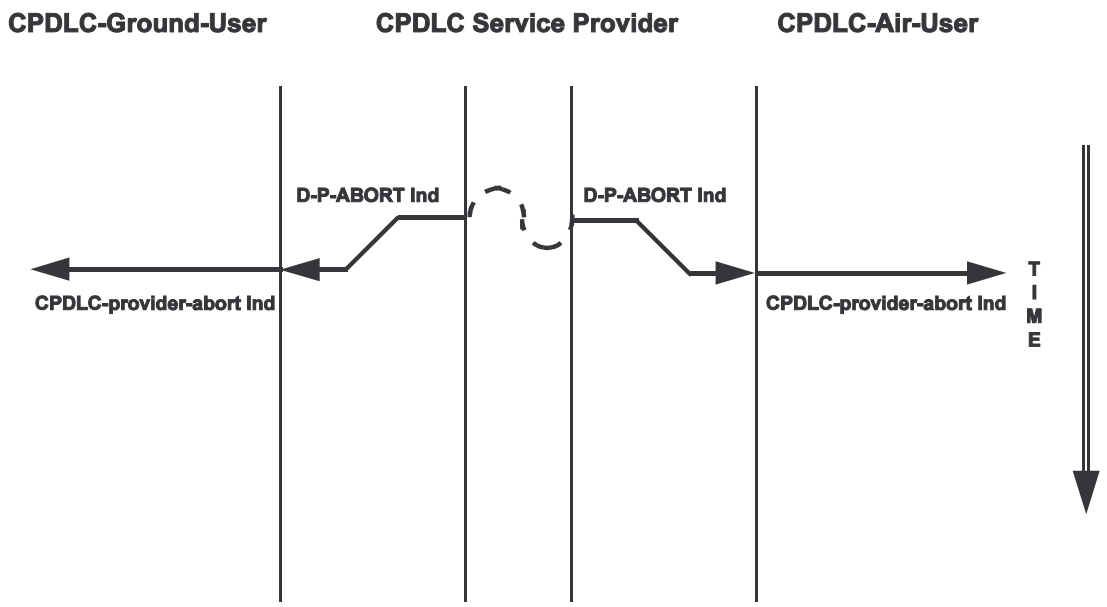


Figure 2.3.5-12. Sequence Diagram for CPDLC-provider-abort Service/Dialogue Service Abort

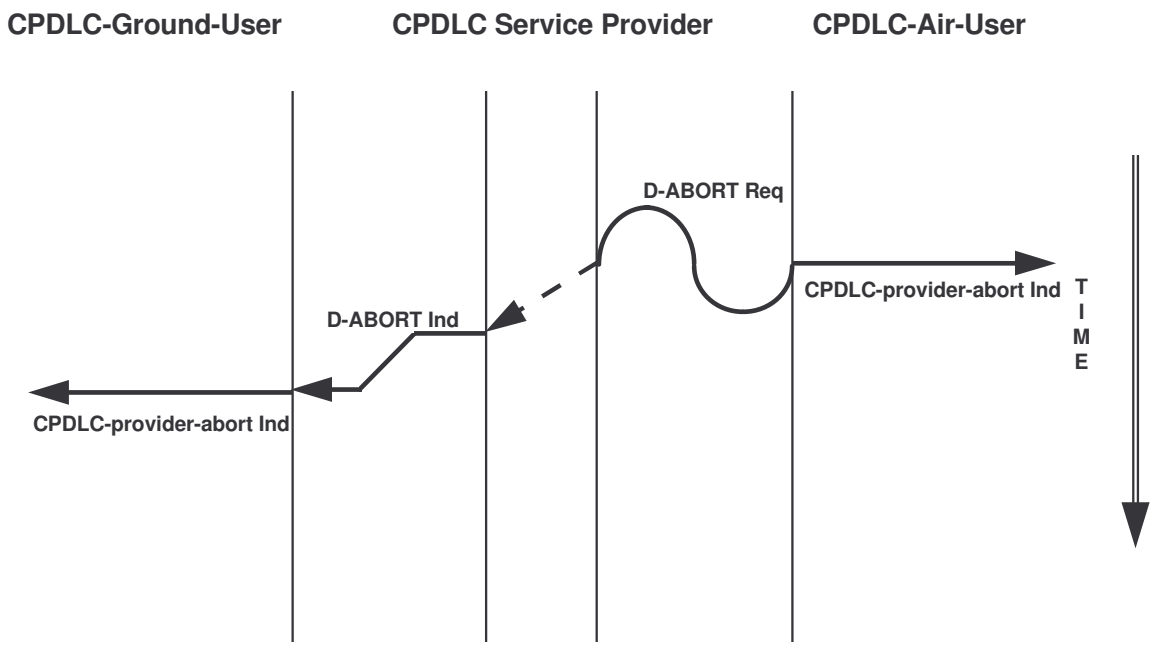


Figure 2.3.5-13. Sequence Diagram for CPDLC-provider-abort Service/CPDLC-Air-ASE Abort

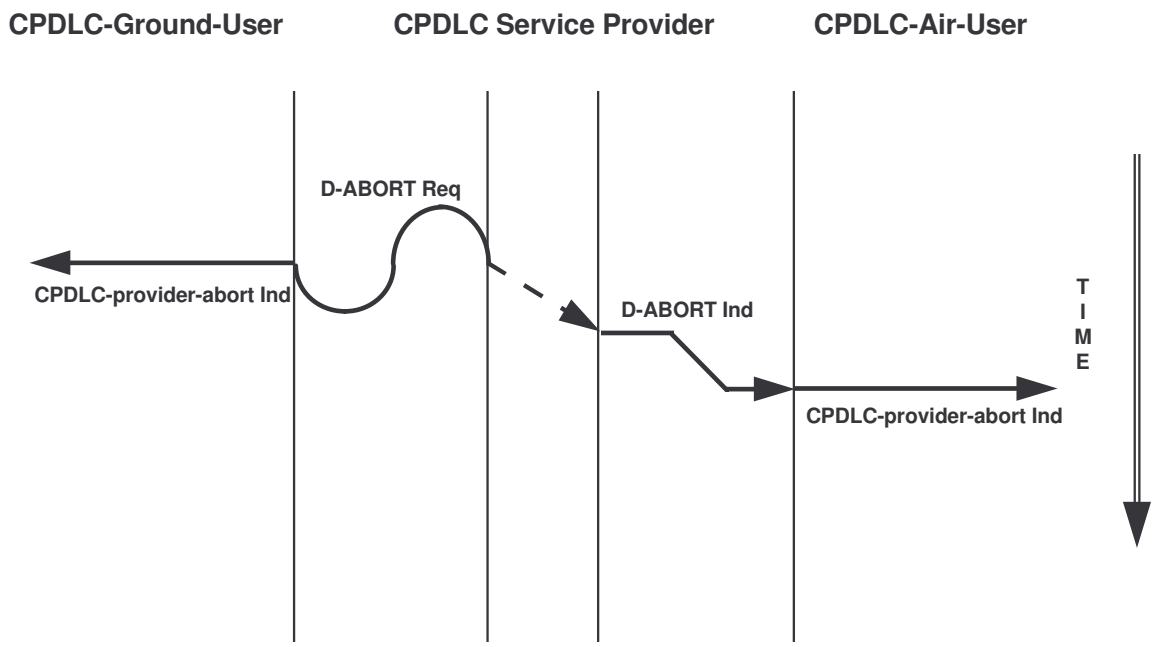


Figure 2.3.5-14. Sequence Diagram for CPDLC-provider-abort Service/ CPDLC-Ground-ASE Abort

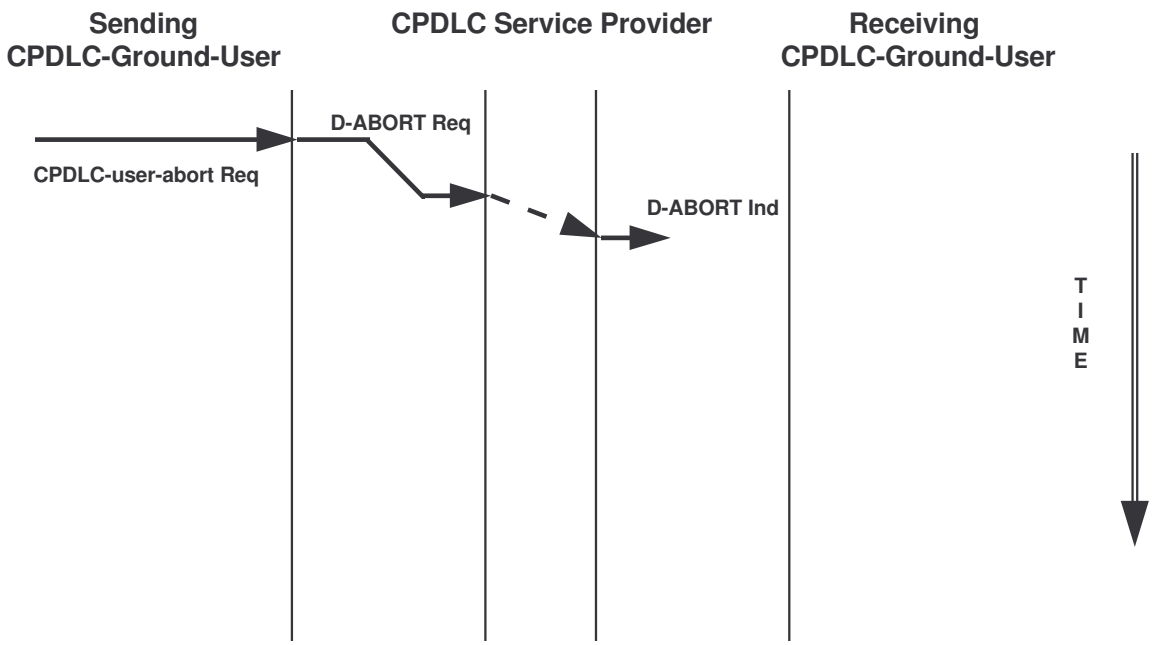


Figure 2.3.5-15. Sequence Diagram for CPDLC-user-abort Service/Sending CPDLC-Ground-User Initiated

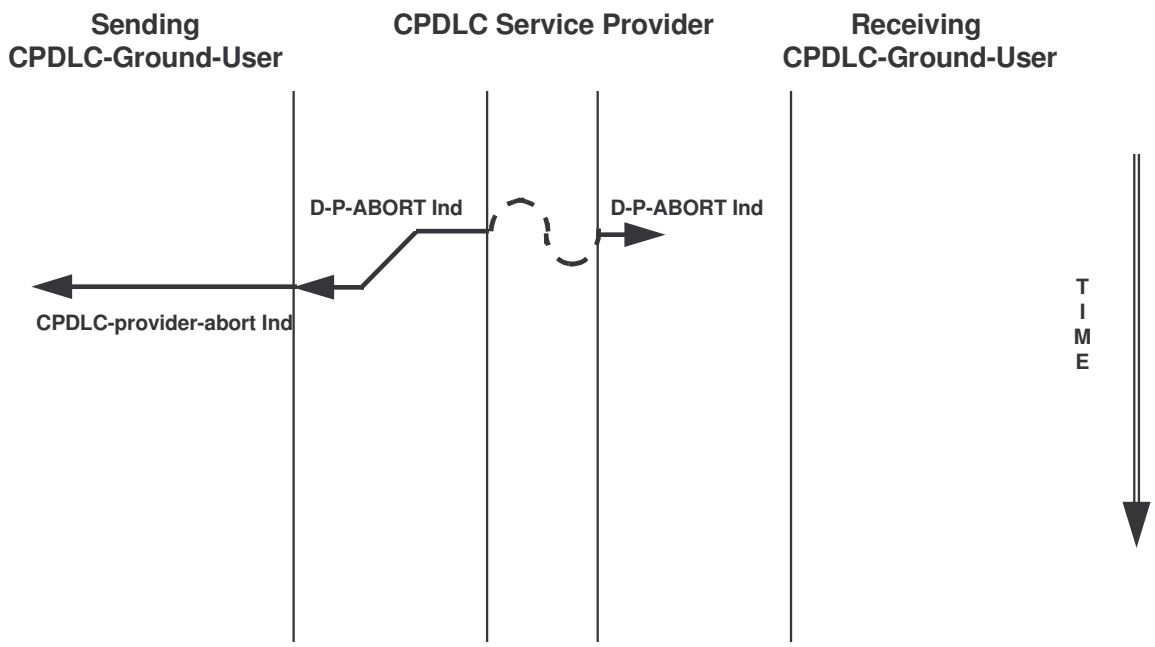


Figure 2.3.5-16. Sequence Diagram for CPDLC-provider-abort Service/Dialogue Service Abort

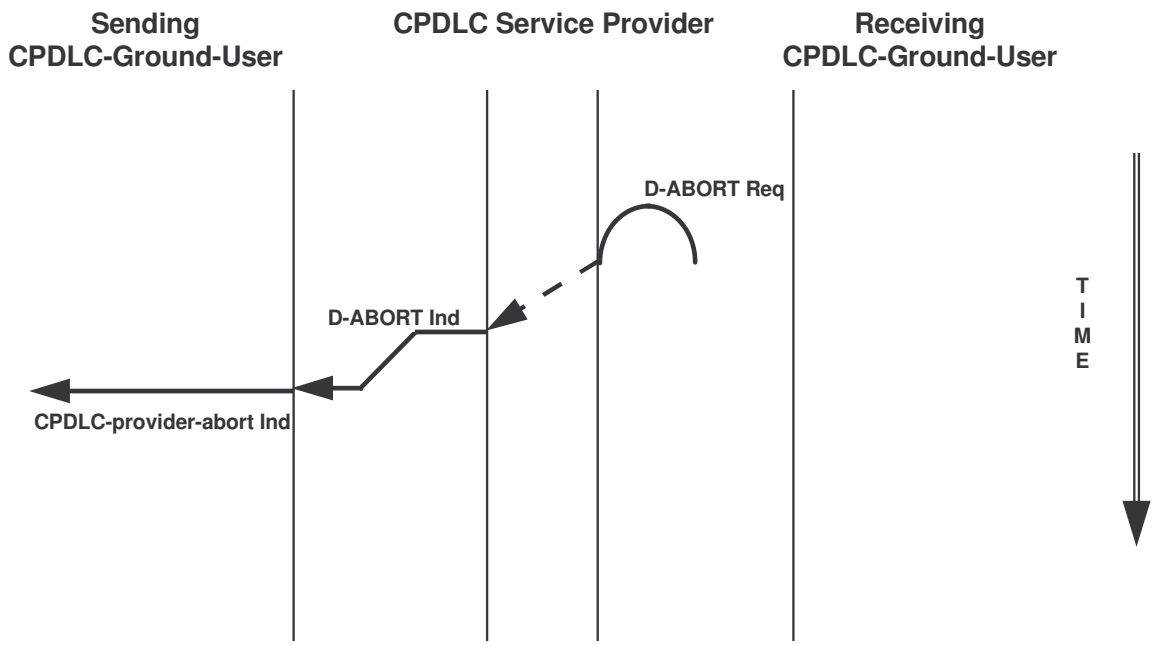


Figure 2.3.5-17. Sequence Diagram for CPDLC-provider-abort Service/Receiving CPDLC-Ground-ASE Abort

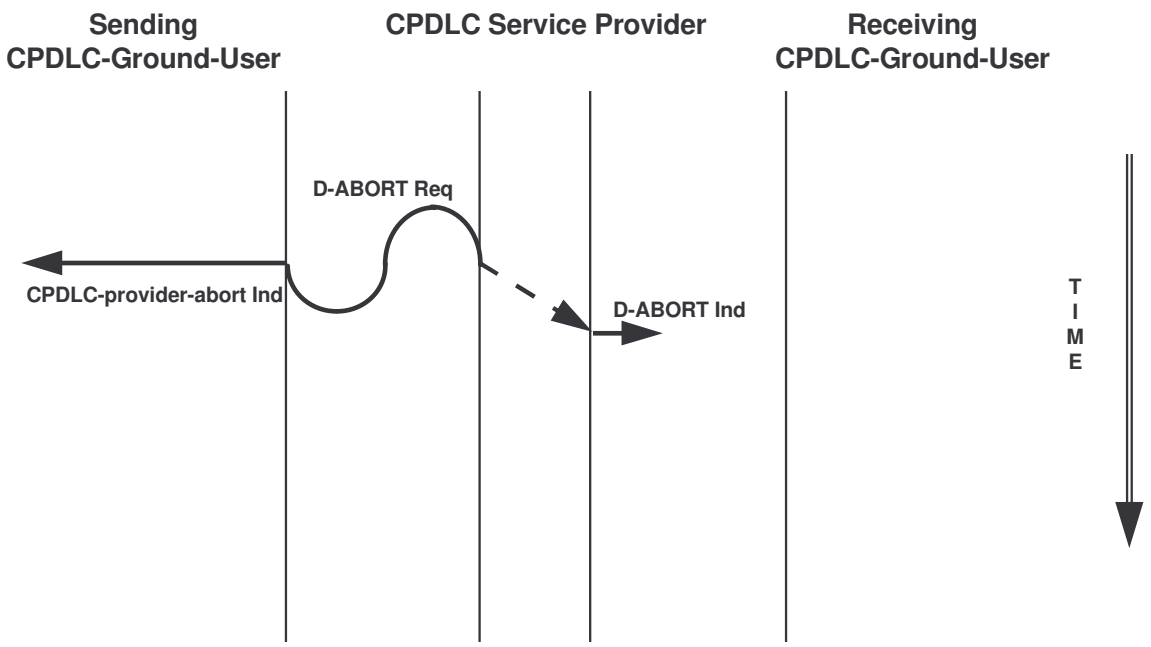


Figure 2.3.5-18. Sequence Diagram for CPDLC-provider-abort Service/Sending CPDLC-Ground-ASE Abort

2.3.5.2 CPDLC Service Provider Timers

2.3.5.2.1 A CPDLC-ASE shall be capable of detecting when a timer expires.

Note 1.— Table 2.3.5-1 lists the time constraints related to the CPDLC application. Each time constraint requires a timer to be set in the CPDLC protocol machine.

Note 2.— If the timer expires before the final event has occurred, a CPDLC-ASE takes appropriate action as defined in 2.3.5.4.1.

2.3.5.2.2 **Recommendation.** — *The timer values should be as indicated in Table 2.3.5-1.*

Table 2.3.5-1. CPDLC Service Provider Timers

CPDLC Service	Timer	Timer Value	Timer Start Event	Timer Stop Event
CPDLC-start	t_{start}	6 minutes	D-START request	D-START confirmation
DSC-start	t_{start}	6 minutes	D-START request	D-START confirmation
CPDLC-forward	t_{start}	6 minutes	D-START request	D-START confirmation

Note.— The receipt of CPDLC-user-abort requests, D-ABORT Indications, or D-P-ABORT Indications are also timer stop events.

2.3.5.3 CPDLC-air-ASE Protocol Description

2.3.5.3.1 Introduction

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

2.3.5.3.1.2 Upon receipt of a PDU, if no actions are described for the arrival of that PDU when a CPDLC-air-ASE is in a specific state, then that PDU is considered not permitted, and exception handling procedures as described in 2.3.5.4.4 shall apply.

2.3.5.3.1.3 If a PDU is received that cannot be decoded, then exception handling procedures as described in 2.3.5.4.3 for invalid PDU shall apply.

2.3.5.3.1.4 If a PDU is not received when one is required, then exception handling as described 2.3.5.4.3 shall apply.

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

a) IDLE

b) START-REQ,

c) START-IND,

d) DIALOGUE, and

e) END.

Note 2.— The CPDLC-air-user is an active user from:

a) the time it has invoked the CPDLC-start service request until:

1) receipt of a CPDLC-start service confirmation with Result parameter equal to the abstract value "rejected", or

2) invocation of a CPDLC-end service response with the Result parameter set to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

b) the time it has received the CPDLC-start service indication until:

1) invocation of a CPDLC-start service response with Result parameter equal to the abstract value "rejected", or

2) invocation of a CPDLC-end service response with the Result parameter set to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

c) the time it has invoked the DSC-start service request until:

1) receipt of a DSC-start service confirmation with Result parameter equal to the abstract value "rejected", or

2) receipt of a DSC-end service confirmation with Result parameter equal to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication.

2.3.5.3.1.5 On initiation the CPDLC-air-ASE shall be in the *IDLE* state.

Note.— The CPDLC-air-ASE contains a Boolean called DSC. DSC has the abstract value "true" when the dialogue is a DSC dialogue, and has the abstract value "false" otherwise.

2.3.5.3.1.6 On the initiation of a CPDLC-air-ASE, DSC shall be set to the abstract value "false".

2.3.5.3.2 D-START Indication

2.3.5.3.2.1 Upon receipt of a D-START indication, if the CPDLC-air-ASE is in the *IDLE* state and the D-START *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 2.3.6-1, 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 (as defined in Annex 10 Volume III Part I Chapter 3), 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 *Protected CPDLC Message* parameter value, and

b) Enter the *START-IND* state.

2.3.5.3.3 D-START Confirmation

2.3.5.3.3.1 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "accepted" and DSC has the abstract value "false" and D-START *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 Message* parameter value, and

2) the abstract value "accepted" as the CPDLC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

2.3.5.3.3.2 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "rejected (permanent)" and the D-START *Reject Source* parameter has the abstract value "DS user" and DSC has the abstract value "false" and the D-START *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 Message* parameter value, and

2) the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.3.3.3 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true" and D-START *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 Message* parameter value, and

2) the abstract value "accepted" as the DSC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

2.3.5.3.3.4 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "rejected (permanent)" and the D-START *Reject Source* parameter has the abstract value "DS user", and DSC has the abstract value "true", andthe D-START *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 Message* parameter value, and

2) the abstract value "rejected" as the DSC-start service *Result* parameter value,

c) Set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.3.4 D-DATA Indication

2.3.5.3.4.1 Upon receipt of a D-DATA indication, if the CPDLC-air-ASE is in the *DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a GroundPDUs [ProtectedUplinkMessage(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 *Protected* CPDLC Message parameter value, and

b) Remain in the *DIALOGUE* state.

2.3.5.3.4.2 Upon receipt of a D-DATA indication, if the CPDLC-air-ASE is in the *END* state and DSC has the abstract value of "true" and the APDU contained in the D-DATA *User Data* parameter is a GroundPDUs [ProtectedUplinkMessage(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 *Protected* CPDLC Message parameter value, and

b) Remain in the *END* state.

2.3.5.3.5 D-END Indication

2.3.5.3.5.1 Upon receipt of a D-END indication, if the CPDLC-air-ASE is in the *DIALOGUE* state, and DSC has the abstract value "false", and the D-END *User Data* parametercontains a GroundPDUs [ProtectedUplinkMessage(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 *Protected* CPDLC Message parameter value, and

b) Enter the *END* state.

2.3.5.3.6 D-END Confirmation

2.3.5.3.6.1 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the abstract the D-END *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true" and the D-END *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 *Protected* CPDLC Message parameter value, and

2) the abstract value "accepted" as the CPDLC-end service *Result* parameter value,

b) Set DSC to the abstract value "false", and

c) Enter the *IDLE* state.

2.3.5.3.6.2 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the D-END *Result* parameter has the abstract value "rejected", and DSC has the abstract value "true", and the D-END *User Data* parameter contains a GroundPDUs [ProtectedUplinkMessage(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 *Protected* CPDLC Message parameter value, and

2) the abstract value "rejected" as the CPDLC-end service *Result* parameter value,

b) Enter the *DIALOGUE* state.

2.3.5.3.7 CPDLC-start Service Request

2.3.5.3.7.1 Upon receipt of a CPDLC-start service request, if the CPDLC-air-ASE is in the *IDLE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a StartDownMessage APDU element containing:

1) the abstract value "cpdlc" as the mode,

2) the *Protected* CPDLC Message parameter as the ProtectedDownlinkMessage,

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.

2.3.5.3.8 CPDLC-start Service Response

2.3.5.3.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-air-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "accepted", and DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a ProtectedDownlinkMessage APDU element based on the *Response Message* 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

2.3.5.3.8.2 Upon receipt of a CPDLC-start service response, if the CPDLC-air-ASE is in the *START-IND* state, and the CPDLC-start service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a ProtectedDownlinkMessage APDU element based on the *Response Message* 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.

2.3.5.3.9 DSC-start Service Request

2.3.5.3.9.1 Upon receipt of a DSC-start service request, if the CPDLC-air-ASE is in the *IDLE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a StartDownMessage APDU element containing:

1) the abstract value "dsc" as the mode, and

2) the *Protected* CPDLC Message parameter as the ProtectedDownlinkMessage,

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.

2.3.5.3.10 CPDLC-message Service Request

2.3.5.3.10.1 Upon receipt of a CPDLC-message service request and the *Protected CPDLC Message* 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 a ProtectedDownlinkMessage APDU-element based on the CPDLC-message service *Protected* CPDLC Message parameter,

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

c) Remain in the *DIALOGUE* state.

2.3.5.3.10.2 Upon receipt of a CPDLC-message service request and the *Protected CPDLC Message* 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 a ProtectedDownlinkMessage APDU-element based on the CPDLC-message service *Protected* CPDLC Message parameter,

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

c) Remain in the *END* state.

2.3.5.3.11 CPDLC-end Service Response

2.3.5.3.11.1 Upon receipt of a CPDLC-end service response, if the CPDLC-air-ASE is in the END state, and the CPDLC-end service *Result* parameter has the abstract value "accepted" and DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a ProtectedDownlinkMessage APDU-element based on the CPDLC-end service *Protected* CPDLC Message 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.

2.3.5.3.11.2 Upon receipt of a CPDLC-end service response, if the CPDLC-air-ASE is in the *END* state, and the CPDLC-end service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "false", the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a ProtectedDownlinkMessage APDU-element based on the CPDLC-end service *Protected* CPDLC Message 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.

2.3.5.3.12 DSC-end Service Request

2.3.5.3.12.1 Upon receipt of a DSC-end service request, if the CPDLC-air-ASE is in the *DIALOGUE* state and DSC has the abstract value "true", the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with a ProtectedDownlinkMessage APDU-element based on the DSC-end service *Protected* CPDLC Message parameter,

b) Invoke D-END request with the APDU as the D-END *User Data* parameter value, and

c) Enter the *END* state.

2.3.5.3.13 CPDLC-user-abort Service Request

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

a) Stop any timer,

b) If the CPDLC-user-abort service *Reason* parameter is provided, create an AircraftPDUs APDU with a CPDLCUserAbortReason APDU-element based on the CPDLC-user-abort service *Reason* parameter,

c) Else create an AircraftPDUs APDU with a CPDLCUserAbortReason [undefined] APDU-element,

d) Invoke D-ABORT request with the following:

1) the D-ABORT *Originator* parameter set to the abstract value "user", and

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

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.3.14 D-ABORT Indication

2.3.5.3.14.1 Upon receipt of a D-ABORT indication, if the CPDLC-air-ASE is not in the *IDLE* state, and the D-ABORT *Originator* parameter is "user" and the D-ABORT *User Data* parameter contains a GroundPDUs [CPDLCUserAbortReason] APDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-user-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CPDLC-user-abort service *Reason* parameter value,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.3.14.2 Upon receipt of a D-ABORT indication, if the CPDLC-air-ASE is not in the *IDLE* state, and if the D-ABORT *Originator* parameter is "provider" and if the D-ABORT *User Data* parameter is provided, the D-ABORT *User Data* parameter contains a GroundPDUs [CPDLCProviderAbortReason] APDU, the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.3.15 D-P-ABORT Indication

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

a) Stop any timer,

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

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4 CPDLC-air-ASE Exception Handling

2.3.5.4.1 A Timer Expires

2.3.5.4.1.1 If a CPDLC-air-ASE detects that a timer has expired, that CPDLC-air-ASE shall:

a) Interrupt any current activity,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [timer-expired] APDU message element,

c) Invoke D-ABORT request with:

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

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

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “timer-expired” as the CPDLC-provider abort service *Reason* parameter value,

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the *IDLE* state.

2.3.5.4.2 Unrecoverable System Error

2.3.5.4.2.1 **Recommendation.** — *If a CPDLC-air-ASE has an unrecoverable system error, the CPDLC-air-ASE should:*

a) Stop any timer,

b) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [undefined-error] APDU message element,

c) Invoke D-ABORT request with:

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

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

d) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “undefined-error” as the CPDLC-provider abort service Reason parameter value,

e) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and

f) Enter the IDLE state.

2.3.5.4.3 Invalid PDU

2.3.5.4.3.1 If the *User Data* parameter of a D-END confirmation with *Result* parameter set to the abstract value “rejected”, or 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.

2.3.5.4.3.2 If the *User Data* parameter of a D-START confirmation with *Result* set to the abstract value "rejected (permanent)", or a D-END confirmation with *Result* set to the abstract value "accepted", is not a valid PDU then the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "invalid-PDU" ,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4.4 Protocol Error

2.3.5.4.4.1 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 2.3.5.3, 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.

2.3.5.4.4.2 (deleted)

2.3.5.4.4.3 If the *User Data* parameter of a D-END confirmation is a valid PDU, but is not a permitted PDU as defined in 2.3.5.3, the CPDLC-air-ASE shall:

a) If the D-END *Result* parameter is set to the abstract value “rejected”, then

1) Stop any timer,

2) Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [protocol-error] APDU message element,

3) Invoke D-ABORT request with:

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

v) 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.

2.3.5.4.4.4 Upon receipt of a Dialogue service primitive for wich there are no instruction in 2.3.5.3 (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.

2.3.5.4.5 D-START Confirmation *Result* or *Reject Source* Not as Expected

2.3.5.4.5.1 If a D-START confirmation *Result* parameter has the abstract value of "rejected (transient)" or if the *Reject Source* parameter has the abstract value of "DS provider", the CPDLC-air-ASE shall:

a) Stop any timer,

b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-error",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.4.6 D-START Indication *Quality of Service* Not as Expected

2.3.5.4.6.1 If a D-START indication *QOS Priority* parameter does not have the abstract value of "high priority flight safety messages" or if the *QOS Residual Error Rate* parameter does not have the abstract value of "low", or if the *QOS Routing Class* parameter does not have one of the abstract values specified in Table 2.3.6-1, the CPDLC-air-ASE shall:

a) Stop any timer,

b) 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.

2.3.5.4.7 Expected PDU Missing

2.3.5.4.7.1 If the *User Data* parameter of a D-START indication or 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.

2.3.5.4.8 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 (as defined in Annex 10 Volume III Part I Chapter 3).*

2.3.5.4.8.1 Upon receipt of a D-START indication with the *Security Requirements* parameter not consistent with the local security policy (as defined in Annex 10 Volume III Part I Chapter 3) 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.

2.3.5.5 CPDLC-ground-ASE Protocol Description

2.3.5.5.1 Introduction

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

2.3.5.5.1.2 Upon receipt of a PDU, if no actions are described for the arrival of that PDU when a CPDLC-ground-ASE is in a specific state, then that PDU is considered not permitted, and exception handling procedures as described in 2.3.5.6.4 shall apply.

2.3.5.5.1.3 If a PDU is received that cannot be decoded, then exception handling procedures as described in 2.3.5.6.3 for invalid PDU shall apply.

2.3.5.5.1.4 If a PDU is not received when one is required, then exception handling as described in 2.3.5.6.3 shall apply.

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

a) IDLE

b) START-REQ,

c) START-IND,

d) DIALOGUE,

e) END, and

f) FORWARD.

Note 2.— The CPDLC-ground-user is an active user from:

a) the time it has invoked the CPDLC-start service request until:

1) receipt of a CPDLC-start service confirmation with Result parameter equal to the abstract value "rejected", or

2) receipt of a CPDLC-end service confirmation with the Result parameter equal to the abstract value "accepted", or

3) invocation of a CPDLC--user-abort service request, or

4) receipt of a CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

b) the time it has received the CPDLC-start service indication until:

1) invocation of a CPDLC-start service response with Result parameter set to the abstract value "rejected", or

2) receipt of a CPDLC-end service confirmation with the Result parameter equal to the abstract value "accepted", or

3) invocation of a CPDLC-user-abort service request, or

4) receipt of CPDLC-user-abort service indication, or

5) receipt of a CPDLC-provider-abort service indication; or

c) the time it has received the DSC-start service indication until:

1) invocation of a DSC-start service response with Result parameter equal to the abstract value "rejected", or

2) invocation of a CPDLC-user-abort service request, or

3) receipt of CPDLC-user-abort service indication, or

4) receipt of a CPDLC-provider-abort service indication; or

d) the time it has invoked the CPDLC-forward service request until:

1) receipt of a CPDLC-forward service confirmation,

2) invocation of a CPDLC-user-abort service request, or

3) receipt of CPDLC-user-abort service indication, or

4) receipt of a CPDLC-provider-abort service indication.

2.3.5.5.1.5 On initiation the CPDLC-ground-ASE shall be in the *IDLE* state.

Note.— The CPDLC-ground-ASE contains a Boolean called DSC. DSC has the abstract value "true" when the dialogue is a DSC dialogue, and has the abstract value "false" otherwise.

2.3.5.5.1.6 On the initiation of a CPDLC-ground-ASE, DSC shall be set to the abstract value "false".

2.3.5.5.2 D-START Indication

2.3.5.5.2.1 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a 24-bit aircraft address, and the D-START *User Data* parameter contains an AircraftPDUs [StartDownMessage] APDU with the APDU-element mode "cpdlc", and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety messages" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low" and the D-START *QOS Routing Class* parameter has one of the abstract values specified in Table 2.3.6-1 and the D-START *Security Requirements* parameter is consistent with the local security policy (as defined in Annex 10 Volume III Part I Chapter 3), 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 *Protected* CPDLC Message parameter value, and

b) Enter the *START-IND* state.

2.3.5.5.2.2 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a 24-bit aircraft address, and the D-START *User Data* parameter contains an AircraftPDUs [StartDownMessage]APDU with the APDU-element mode "dsc", and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety messages" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low", and the D-START *QOS Routing Class* parameter has one of the abstract values specified in Table 2.3.6-1, and the D-START *Security Requirements* parameter is consistent with the local security policy (as defined in Annex 10 Volume III Part I Chapter 3), 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 *Protected* CPDLC Message parameter value,

b) Set DSC to "true", and

c) Enter the *START-IND* state.

2.3.5.5.2.3 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a Facility Designation, and the D-START *User Data* parameter contains a GroundPDUs [ATCForwardMessage] APDU and the CPDLC-ground-ASE supports the CPDLC-forward service, and the D-START *QOS Priority* parameter has the abstract value "high priority flight safety messages" and the D-START *QOS Residual Error Rate* parameter has the abstract value "low", the CPDLC-ground-ASE shall:

a) If the D-START *DS User Version Number* parameter value is equal to the CPDLC-ground-ASE version number and the D-START *Security Requirements* parameter is consistent with the local security policy (as defined in Annex 10 Volume III Part I Chapter 3):

1) Invoke CPDLC-forward service indication containing the following:

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

ii) set the D-START GroundPDUs APDU-element as the CPDLC-forward service *CPDLC Message* parameter value, and

2) Create a GroundPDUs APDU with an ATCForwardResponse [success] APDU element,

3) Invoke D-START response with the following:

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

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

4) Remain in the *IDLE* state.

b) If the D-START *DS User Version Number* parameter value is not equal to the CPDLC-ground-ASE version number:

1) Create a GroundPDUs APDU with an ATCForwardResponse [version-not-equal] APDU element,

2) Invoke D-START response with the following:

i) the CPDLC-ground-ASE version number as the D-START *DS User Version Number* parameter value,

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

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

3) Remain in the *IDLE* state.

2.3.5.5.2.4 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a Facility Designation, and the D-START *User Data* parameter contains a GroundPDUs APDU and the APDU element is an ATCForwardMessage and the CPDLC-ground-ASE does not support the CPDLC-forward service and the D-START *Security Requirements* parameter is consistent with the local security policy (as defined in Annex 10 Volume III Part I Chapter 3), the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCForwardResponse [service-not-supported] APDU element,

b) Invoke D-START response with the following:

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

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

c) Remain in the *IDLE* state.

2.3.5.5.3 D-START Confirmation

2.3.5.5.3.1 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *START-REQ* state and if the D-START *Result* parameter has the abstract value "accepted", and DSC has the abstract value of "false" and D-START *User Data* parameter contains an AircraftPDUs [ProtectedDownlinkMessage] 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 Message* parameter value, and

2) the abstract value "accepted" as the CPDLC-start service *Result* parameter value, and

c) Enter the *DIALOGUE* state.

2.3.5.5.3.2 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "rejected (permanent)" and the D-START *Reject Source* parameter has the abstract value "DS user" and DSC has the abstract value "false" and the D-START *User Data* parameter contains an AircraftPDUs [ProtectedDownlinkMessage] 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 Message parameter value, and

2) the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and

c) Enter the *IDLE* state.

2.3.5.5.3.3 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *FORWARD* state and if the D-START *Result* parameter has the abstract value "rejected (permanent)" and the *Reject Source* parameter has the abstract value "DS user"and the D-START *User Data* parameter contains a GroundPDUs [ATCForwardResponse] APDU 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.

2.3.5.5.4 D-DATA Indication

2.3.5.5.4.1 Upon receipt of a D-DATA indication, if the CPDLC-ground-ASE is in the *DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a AircraftPDUs [ProtectedDownlinkMessage] 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 *Protected* CPDLC Message parameter value, and

b) Remain in the *DIALOGUE* state.

2.3.5.5.4.2 Upon receipt of a D-DATA indication, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value "false" and the APDU contained in the D-DATA *User Data* parameter is an AircraftPDUs [ProtectedDownlinkMessage] 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 *Protected* CPDLC Message parameter value, and

b) Remain in the *END* state.

2.3.5.5.5 D-END Indication

2.3.5.5.5.1 Upon receipt of a D-END indication, if the CPDLC-ground-ASE is in the *DIALOGUE* state, and DSC has the abstract value "true", and the D-END *User Data* parameter contains an AircraftPDUs [ProtectedDownlinkMessage] 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 *Protected* CPDLC Message parameter value, and

b) Enter the *END* state.

2.3.5.5.6 D-END Confirmation

2.3.5.5.6.1 Upon receipt of a D-END confirmation, if the CPDLC-ground-ASE is in the *END* state and the D-END *Result* parameter has the abstract value "accepted" and DSC has the abstract value "false" and the D-END *User Data* parameter contains an AircraftPDUs [ProtectedDownlinkMessage] 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 *Protected* CPDLC Message parameter value, and

2) The abstract value "accepted" as the CPDLC-end service *Result* parameter value, and

b) Enter the *IDLE* state.

2.3.5.5.6.2 Upon receipt of a D-END confirmation, if the CPDLC-ground-ASE is in the *END* state and the D-END *Result* has the abstract value "rejected" and DSC has the abstract value "false", and the D-END *User Data* parameter contains an AircraftPDUs [ProtectedDownlinkMessage] 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 *Protected* CPDLC Message parameter value, and

2) The abstract value "rejected" as the CPDLC-end service *Result* parameter value, and

b) Enter the *DIALOGUE* state.

2.3.5.5.7 CPDLC-start Service Request

2.3.5.5.7.1 Upon receipt of a CPDLC-start service request, if the CPDLC-ground-ASE is in the *IDLE* state, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an UplinkMessage(startup) APDU element containing the *Protected* CPDLC Message 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.

2.3.5.5.8 CPDLC-start Service Response

2.3.5.5.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "accepted" , and DSC has the abstract value "false", the CPDLC-ground-ASE shall:

a) create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the *Response Message* 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.

2.3.5.5.8.2 Upon receipt of a CPDLC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "false", the CPDLC-ground-ASE shall:

a) create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the *Response Message* 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.

2.3.5.5.9 DSC-start Service Response

2.3.5.5.9.1 Upon receipt of a DSC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the DSC-start service *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true", the CPDLC-ground-ASE shall:

a) create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the *Response Message* 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.

2.3.5.5.9.2 Upon receipt of a DSC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the DSC-start service *Result* parameter has the abstract value "rejected" and DSC has the abstract value "true", the CPDLC-ground-ASE shall:

a) create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the *Response Message* 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.

2.3.5.5.10 CPDLC-message Service Request

2.3.5.5.10.1 Upon receipt of a CPDLC-message service request and the *Protected CPDLC Message* 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 a ProtectedUplinkMessage(send) APDU element based on the CPDLC-message service *Protected* CPDLC Message parameter,

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

c) Remain in the *DIALOGUE* state.

2.3.5.5.10.2 Upon receipt of a CPDLC-message service request and the *Protected CPDLC Message* 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 a ProtectedUplinkMessage(send) APDU element based on the CPDLC-message service *Protected* CPDLC Message parameter,

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

c) Remain in the *END* state.

2.3.5.5.11 CPDLC-end Service Request

2.3.5.5.11.1 Upon receipt of a CPDLC-end service request, if the CPDLC-ground-ASE is in the *DIALOGUE* state and DSC has the abstract value "false", the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the CPDLC-end service *Protected* CPDLC Message parameter,

b) Invoke D-END request with the APDU as the D-END *User Data* parameter value, and,

c) Enter the *END* state.

2.3.5.5.12 DSC-end Service Response

2.3.5.5.12.1 Upon receipt of a DSC-end service response, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value "true", and the DSC-end service *Result* parameter has the abstract value "accepted", the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the DSC-end service *Protected CPDLC Message* 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.

2.3.5.5.12.2 Upon receipt of a DSC-end service response, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value "true", and the DSC-end service *Result* parameter has the abstract value "rejected", the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with a ProtectedUplinkMessage(send) APDU element based on the DSC-end service *Protected CPDLC Message* 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.

2.3.5.5.13 CPDLC-forward Service Request

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

a) Create a GroundPDUs APDU with an ATCForwardMessage APDU element based on the CPDLC-forward service *CPDLC Message* parameter,

b) Invoke D-START request with the following:

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

2) the CPDLC-start service *Calling Facility Designation* parameter value as the D-START *Calling Peer ID* parameter value,

3) the 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.

2.3.5.5.14 CPDLC-user-abort Service Request

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

a) Stop any timer,

b) If the CPDLC-user-abort service *Reason* parameter is provided, create a GroundPDUs APDU with a CPDLCUserAbortReason APDU element based on the CPDLC-user-abort service *Reason* parameter,

c) Else create a GroundPDUs APDU with a CPDLCUserAbortReason [undefined] APDU element,

d) Invoke D-ABORT request with the following:

1) the D-ABORT *Originator* parameter set to the abstract value "user", and

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

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.5.15 D-ABORT Indication

2.3.5.5.15.1 Upon receipt of a D-ABORT indication, if the CPDLC-ground-ASE is not in the *IDLE* state and the D-ABORT *Originator* parameter is "user" and the D-ABORT *User Data* parameter contains an AircraftPDUs [CPDLCUserAbortReason] APDU, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-Ground-User is an active user, invoke CPDLC-user-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CPDLC-user-abort service *Reason* parameter value,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.5.15.2 Upon receipt of a D-ABORT indication, if the CPDLC-ground-ASE is not in the *IDLE* state, and if the D-ABORT *Originator* parameter is "provider" and if the D-ABORT *User Data* parameter is provided, the D-ABORT *User Data* parameter contains either an AircraftPDUs [CPDLCProviderAbortReason] APDU or a GroundPDUs [CPDLCProviderAbortReason] APDU , the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-Ground-User is an active user, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided,

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.5.16 D-P-ABORT Indication

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

a) Stop any timer,

b) If the CPDLC-Ground-User is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-failure",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.6 CPDLC-ground-ASE Exception Handling

2.3.5.6.1 A Timer Expires

2.3.5.6.1.1 If a CPDLC-ground-ASE detects that a timer has expired, that CPDLC-ground-ASE shall:

a) Interrupt any current activity,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [timer-expired] APDU message element,

c) Invoke D-ABORT request with:

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

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

d) If the CPDLC-Ground-User is an active user, invoke CPDLC-provider-abort service indication with the abstract value "timer-expired" as the CPDLC-provider abort service *Reason* parameter value",

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the *IDLE* state.

2.3.5.6.2 Unrecoverable System Error

2.3.5.6.2.1 **Recommendation.** — *If a CPDLC-ground-ASE has an unrecoverable system error, the CPDLC-ground-ASE should:*

a) Stop any timer,

b) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [undefined-error] APDU message element,

c) Invoke D-ABORT request with:

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

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

d) If the CPDLC-Ground-User is an active user, invoke CPDLC-provider-abort service indication with the abstract value "undefined-error" as the CPDLC-provider abort service Reason parameter value",

e) If DSC has the abstract value "true", set DSC to the abstract value "false", and

f) Enter the IDLE state.

2.3.5.6.3 Invalid PDU

2.3.5.6.3.1 If the *User Data* parameter of a D-END confirmation with *Result* parameter set to the abstract value "rejected", 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.

2.3.5.6.3.2 If the *User Data* parameter of a D-START confirmation with *Result* set to the abstract value "rejected (permanent)", or a D-END confirmation with *Result* set to the abstract value "accepted", is not a valid PDU the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the CPDLC-Ground-User is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "invalid-PDU",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.6.4 Protocol Error

2.3.5.6.4.1 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 2.3.5.5, 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.

2.3.5.6.4.2 (deleted)

2.3.5.6.4.3 If the *User Data* parameter of a D-END confirmation is a valid PDU, but is not a permitted PDU for which action is described within a given state as defined in 2.3.5.5, the CPDLC-ground-ASE shall:

a) Stop any timer,

b) If the D-END *Result* parameter is set to the abstract value "rejected", then

1) Create a GroundPDUs APDU with a CPDLCProviderAbortReason [protocol-error] APDU message element,

2) Invoke D-ABORT request with:

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

v) 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.

2.3.5.6.4.4 Upon receipt of a Dialogue service primitive for wich there are no instruction in 2.3.5.3 (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.

2.3.5.6.5 D-START Confirmation Result or Reject Source Not as Expected

2.3.5.6.5.1 If a D-START confirmation *Result* parameter has the abstract value of "rejected (transient)" or if the *Reject Source* parameter has the abstract value of "DS provider", the CPDLC-ground-ASE shall:

a) Stop any timer,

b) if the CPDLC-Ground-User is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-error",

c) If DSC has the abstract value "true", set DSC to the abstract value "false", and

d) Enter the *IDLE* state.

2.3.5.6.6 D-START Indication Quality of Service Not as Expected

2.3.5.6.6.1 If a D-START indication *QOS Priority* parameter does not have the abstract value of "high priority flight safety messages" or if the *QOS Residual Error Rate* parameter does not have the abstract value of "low", or if the *QOS Routing Class* parameter does not have one of the abstract values specified in Table 2.3.6-1, the CPDLC-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.

2.3.5.6.7 Expected PDU Missing

2.3.5.6.7.1 If the *User Data* parameter of a D-START indication or 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.

2.3.5.6.8 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 (as defined in Annex 10 Volume III Part I Chapter 3).*

2.3.5.6.8.1 Upon receipt of a D-START indication with the *Security Requirements* parameter not consistent with the local security policy (as defined in Annex 10 Volume III Part I Chapter 3) 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.

2.3.5.7 CPDLC ASE State Tables

2.3.5.7.1 Priority

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

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

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

Table 2.3.5-2. CPDLC-air-ASE State Table

STATE ⇒ **EVENT ⇓**	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
Dialogue Service Events					
D-START Indication APDU = ProtectedUplinkMessage (startup)	● CPDLC-start indication ⇒*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "false" APDU = ProtectedUplinkMessage (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 = ProtectedUplinkMessage (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 = ProtectedUplinkMessage (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 = ProtectedUplinkMessage (send)	cannot occur	●Stop timer t_{start} ● DSC-start confirmation ●Set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur
D-DATA Indication APDU = ProtectedUplinkMessage (send)	cannot occur	cannot occur	cannot occur	●*CPDLC*-message indication ⇒*DIALOGUE*	●if DSC="true" ● CPDLC-message indication ⇒*END* ●else cannot occur
D-END Indication: DSC="false" APDU = ProtectedUplinkMessage (send)	cannot occur	cannot occur	cannot occur	● CPDLC-end indication ⇒*END*	cannot occur

STATE ➡ EVENT ⬇	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
D-END Confirmation: DSC="true" *Result* "accepted" APDU = ProtectedUplinkMessage (send)	cannot occur	cannot occur	cannot occur	cannot occur	● DSC-end confirmation ●Set DSC "false" ⇒*IDLE*
D-END Confirmation: DSC="true", *Result* "rejected" APDU = ProtectedUplinkMessage (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-abor t indication ●If DSC = "true", set DSC = "false" ⇒*IDLE*
D-ABORT Indication *Originator* "provider"	cannot occur	●Stop timer T_{start}, if set ●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC "false" ⇒*IDLE*	●Stop timer T_{start}, if set ●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC "false" ⇒*IDLE*	●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC "false" ⇒*IDLE*	●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC "false" ⇒*IDLE*
D-P-ABORT indication	cannot occur	●Stop timer t_{start}, if set ●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC = "false" ⇒*IDLE*	●Stop timer t_{start}, if set ●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC = "false" ⇒*IDLE*	●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC = "false" ⇒*IDLE*	●If active user: CPDLC-provider-abort indication ●If DSC = "true", set DSC = "false" ⇒*IDLE*
T_{start} Expires	cannot occur	●D-ABORT request ● CPDLC-provider-abort indication ●If DSC = "true", set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur

Table 2.3.5-3. CPDLC-ground-ASE State Table

STATE ➡ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
Dialogue Service Events						
D-START Indication APDU Aircraft mode "cpdlc"	● CPDLC-start indication ⇒*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication APDU Aircraft mode "dsc"	● DSC-start indication, ●Set DSC ="true" ⇒*START-IND*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication APDU Forward version equal and function supported	● CPDLC-forward indication ●D-START response ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Indication APDU Forward version not equal or function not supported	●D-START response ⇒*IDLE*	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted" DSC = "false"	cannot occur	●Stop timer t_{start} ● CPDLC-start confirmation ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user" DSC="false"	cannot occur	●Stop timer t_{start} ● CPDLC-start confirmation ⇒*IDLE*	cannot occur	cannot occur	cannot occur	●Stop timer t_{start} ●CPDLC-forward confirmation ⇒*IDLE*
D-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*

2.3.6 COMMUNICATION REQUIREMENTS

2.3.6.1 Encoding Rules

2.3.6.1.1 The CPDLC application shall use PER as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 CPDLC message structure and content specified in 2.3.4.

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.

2.3.6.2 Dialogue Service Requirements

2.3.6.2.1 Primitive Requirements

2.3.6.2.1.1 Where dialogue service primitives, that is D-START, D-DATA, D-END, D-ABORT, and D-P-ABORT are described as being invoked in 2.3.5, the CPDLC-ground-ASE and the CPDLC-air-ASE shall exhibit external behavior consistent with the dialogue service, as described in 4.2 having been implemented and its primitives invoked.

2.3.6.2.2 Quality-of-Service Requirements

2.3.6.2.2.1 The application service priority for CPDLC shall have the abstract value of “high priority flight safety messages”.

2.3.6.2.2.2 The RER Quality of Service Parameter of the D-START shall be set to the abstract value of “low”.

2.3.6.2.2.3 The CPDLC-ASE shall map the CPDLC-start service or DSC-start service *Class of Communication* parameter abstract value to the ATSC routing class abstract value part of the D-START QOS parameter as presented in Table 2.3.6-1.

Table 2.3.6-1. Mapping Between Class of Communication and Routing Class Abstract Values

Class of Communication Abstract Value	Routing Class Abstract Value
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 1.3.

2.3.6.2.3 ATN Security Requirements

2.3.6.2.3.1 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.

2.3.7 CPDLC USER REQUIREMENTS

2.3.7.1 General

Note 1.— Requirements imposed on CPDLC-user concerning Protected CPDLC messages and CPDLC messages and interfacing with the CPDLC-ASEs are presented in this chapter.

Note 2.— Where reference is made to the "CPDLC-user", this implies both the CPDLC-air-user and the CPDLC-ground-user.

Note 3.— A "Protected CPDLC Message" is:

a) what the CPDLC-user provides the CPDLC-service as the Protected CPDLC Message or Response Message 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 a Protected CPDLC Message.

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 Protected CPDLC Message has been provided by the CPDLC-service.

Note 6.—A CPDLC-air-user uses the contents of an uplink Protected CPDLC Message 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 Protected CPDLC Message 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 Application Message Integrity Check generation/validation algorithm other than the ATN Message Checksum specified in 4.10.2 is out of scope of this specification and requires prior agreement by both CPDLC air and ground users.

2.3.7.1.1 General CPDLC-service Requirements

2.3.7.1.1.1 A CPDLC-ground-user shall invoke CPDLC-start service, DSC-start service, CPDLC-message service, CPDLC-end service, and DSC-end service only when communicating with a CPDLC-air-user.

2.3.7.1.1.2 A CPDLC-ground-user shall invoke CPDLC-forward service, only when communicating with another CPDLC-ground-user.

Note 1.— When a CPDLC-user invokes the CPDLC-start service, the DSC-start service, or the CPDLC-forward service and requires a particular class of communication service, the CPDLC-user sets the Class of Communication Service parameter.

Note 2.— When a CPDLC-user does not require a particular class of communication, the user does not set the Class of Communication Service parameter.

Note 3.— When the CPDLC-user requires secure or unsecure services, the user sets the Security Required parameter as appropriate.

2.3.7.2 Protected CPDLC Message Generation Requirements

Note.—When a CPDLC-user is required to send a CPDLC Message within the Protected CPDLC Message parameter of a CPDLC Start, a DSC Start, a CPDLC Message, a CPDLC End, or a DSC End, it is implied that the CPDLC Message is encoded within a ProtectedUplinkMessage or ProtectedDownlinkMessage as specified in this section.

2.3.7.2.1 Downlink Protected CPDLC Message

2.3.7.2.1.1 The CPDLC-air-user shall compose a downlink Protected CPDLC Message from:

a) an optional identifier for the algorithm used to compute the Application Message Integrity Check.,

b) an optional downlink CPDLC Message, and

c) an Application Message 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 ContentProtectedMessage.

Note 2.—The Application Message Integrity Check can be generated even when the CPDLC Message is not present. With the default Application Message Integrity Check, this still permits the sender and receiver to be verified.

2.3.7.2.1.2 Unless it is has been agreed by means outside of this specification that an alternative Application Message Integrity Check generation algorithm is used, the CPDLC-air-user shall compute the Application Message Integrity Check using the default *Application Message Integrity Check* generation algorithm.

2.3.7.2.1.3 The default Application Message Integrity Check generation algorithm shall be the computation of the Application Message Integrity Check using the default ATN Message Checksum specified in 4.10.2 over the ContentProtectedMessage specified in 2.3.7.3.2.

2.3.7.2.1.4 When a downlink CPDLC Message is provided in a *Protected CPDLC Message* or *Response Message* parameter, it shall be created by the CPDLC-air-user from an ATCDownlinkMessage as defined in 2.3.4.3, 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 lenght determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

2.3.7.2.1.5 When the value of the Application Message Integrity Check is computed using the default *Application Message Integrity Check* generation algorithm specified in 4.10.2, the CPDLC-air-user shall omit the algorithm identifier on the first downlink Protected CPDLC Message or set it to the algorithm identifier atn-default-checksum.

2.3.7.2.1.6 When an Application Message Integrity Check generation algorithm other than the default is used, the CPDLC-air-user shall set the algorithm identifier on the first downlink Protected CPDLC Message to the value of the relative OID that identifies the Application Message Integrity Check generation algorithm used.

2.3.7.2.1.7 In subsequent downlink Protected CPDLC Messages, the CPDLC-air-user shall omit the algorithm identifier.

2.3.7.2.2 Uplink Protected CPDLC Message

2.3.7.2.2.1 The CPDLC-ground-user shall compose an uplink Protected CPDLC Message from:

a) an optional identifier for the algorithm used to compute the Application Message Integrity Check,

b) an optional uplink CPDLC Message and

c) an Application Message Integrity Check.

Note.— The Application Message Integrity Check can be generated even when the CPDLC Message is not present. With the default Application Message Integrity Check, this still permits the sender and receiver to be verified.

2.3.7.2.2.2 Unless it is has been agreed by means outside of this specification that an alternative Application Message Integrity Check algorithm is used, the CPDLC-ground-user shall compute the Application Message Integrity Check using the default *Application Message Integrity Check generation algorithm* specified in 4.10.2.

2.3.7.2.2.3 When an uplink CPDLC Message is provided in a *Protected CPDLC Message* or *Response Message* parameter, it shall be created by the CPDLC-ground-user from an ATCUplinkMessage as defined in 2.3.4.3, 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 lenght determinant includes only the significant bits of the encoding, corresponding to the ASN.1 type.

2.3.7.2.2.4 When the value of the Application Message Integrity Check is computed using the default Application Message Integrity Check generation algorithm generation algorithm specified in 4.10.2, the CPDLC-ground-user shall omit the algorithm identifier on the first uplink Protected CPDLC Message or set it to the algorithm identifier atn-default-checksum.

2.3.7.2.2.5 When an Application Message Integrity Check generation algorithm other than the default is used, the CPDLC-ground-user shall set the algorithm identifier on the first uplink Protected CPDLC Message to the value of the relative OID that identifies the Application Message Integrity Check generation algorithm used.

2.3.7.2.2.6 In subsequent uplink Protected CPDLC Messages, the CPDLC-ground-user shall omit the algorithm identifier.

2.3.7.3 The Application Message Integrity Check

2.3.7.3.1 The message to be used by the sending CPDLC-user to generate the Application Message Integrity Check shall consist of the unaligned basic PER encoding of the ASN.1 ContentProtectedMessage data type, created from:

a) The aircraft Flight Identification assigned to the CPDLC-air-user, expressed in canonical form as defined in 2.3.7.3.4,

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 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 ContentProtectedMessage is created solely for the purposes of Application Message Integrity Check computation and is never exchanged over a CPDLC or DSC dialogue.

2.3.7.3.2 The ContentProtectedMessage data type shall comply with the ASN.1 module ATCMessageIntegrityCheckVersion1, as defined in this section.

```
ATCMessageIntegrityCheckVersion1 DEFINITIONS ::=

BEGIN
```

```
IMPORTS
        AircraftFlightIdentification,
        AircraftAddress,
        FacilityDesignation FROM CPDLCMessageSetVersion1;

ContentProtectedMessage ::= SEQUENCE
        {
                flightID                        AircraftFlightIdentification,
                aircraftAddress                 AircraftAddress,
                facilityDesignator              FacilityDesignation,
                cPDLCMessageAbstractSyntax      OBJECT IDENTIFIER,
                protectedMessage                CPDLCMessage OPTIONAL
        }

CPDLCMessage ::= BIT STRING

END
```

2.3.7.3.3 Aircraft Flight Identification

Note.—The Application Message 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).

2.3.7.3.3.1 The Aircraft Flight Identification (FlightID) shall be the Aircraft Identification as given in Field 7 of the filed Flight Plan.

2.3.7.3.3.2 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.

2.3.7.3.3.3 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.

2.3.7.4 CPDLC Message Generation Requirements

Note 1.—A response message is a message which is a reply to a received message. It contains a message reference number identical to the message identification number of the message to which it refers. Only response messages contain a message reference number.

Note 2.— Message response attributes dictate a) if a response is required or prohibited; b) if a response is required, dictate the permitted response messages.

Note 3.— A closure response message is a reply to a message or series of messages which terminates a sequence of message exchanges. However due to the multiple element capability of a CPDLC Message, a closure message may contain message element(s) in addition to the required closure message element that initiate a new sequence of messages.

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 protectedMessage element of a Protected CPDLC Message.

2.3.7.4.1 Message Composition

2.3.7.4.1.1 A CPDLC Message shall be composed of a message header, and from one to five message elements.

2.3.7.4.1.2 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 acknowledgment requirement (optional).

2.3.7.4.1.3 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.

2.3.7.4.1.4 A message element shall consist of a message element identifier, data as indicated by the specified message element, and associated message element attributes.

2.3.7.4.1.5 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 message,

c) date and time,

d) a logical acknowledgment indication, if required,

e) from one to five message element identifiers, and

f) data as required for each message element identification included.

2.3.7.4.1.6 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.

2.3.7.4.2 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.

2.3.7.4.2.1 The message identification number provided by the CPDLC-user shall be different from any other message identification number currently in use.

2.3.7.4.2.2 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.

2.3.7.4.2.3 If a CPDLC or DSC dialogue is terminated, all message identification numbers pertaining to that dialogue shall be considered available.

2.3.7.4.3 Message Elements That Cannot Be Combined With Other Message Elements in a Message

2.3.7.4.3.1 The LOGICAL ACKNOWLEDGMENT message element (uplink message element 227 and downlink message element 100) shall be sent only as a single message element CPDLC Message.

2.3.7.4.3.2 The NEXT DATA AUTHORITY [facility] message element (uplink message element 160) shall be sent only as a single message element CPDLC Message.

2.3.7.4.4 Restriction on Route Clearance Variable Message Elements

2.3.7.4.4.1 A CPDLC Message shall contain no more than two message elements with the [routeClearance] variable.

2.3.7.5 Protected CPDLC Message Receipt Requirements

2.3.7.5.1 Downlink Protected CPDLC Message

2.3.7.5.1.1 On receipt of a downlink Protected CPDLC Message, the Application Message Integrity Check verification procedure shall be performed before any other CPDLC Message processing.

2.3.7.5.1.2 On receipt of the first downlink Protected CPDLC Message, the CPDLC-ground-user shall verify the Application Message Integrity Check verification procedure using:

a) the ATN Message Checksum verification procedure specified in 4.10.2 if the downlink Protected CPDLC Message does not contain an algorithm identifier or contains the algorithm identifier atn-default-checksum,

b) the specified algorithm if the downlink Protected CPDLC Message contains any other algorithm identifier.

2.3.7.5.1.3 On receipt of a subsequent downlink Protected CPDLC Message, the CPDLC-ground-user shall:

a) ignore the specified algorithm if the downlink Protected CPDLC Message contains an algorithm identifier,

b) verify the Application Message Integrity Check using the Application Message Integrity Check verification procedure used for the preceding downlink Protected CPDLC Message.

2.3.7.5.1.4 On receipt of a downlink Protected CPDLC Message containing an unrecognised Application Message Integrity Check algorithm identifier, the CPDLC-ground-user shall:

a) discard the received downlink Protected CPDLC Message, 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 Protected CPDLC Message that includes an unrecognised algorithm identifier.

2.3.7.5.1.5 When the Application Message Integrity Check verification procedure fails, the CPDLC-ground-user shall:

a) discard the received downlink Protected CPDLC Message, 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 Protected CPDLC Message and the Application Message Integrity Check verification procedure fails.

2.3.7.5.1.6 If the Application Message Integrity Check verification procedure of a downlink Protected CPDLC Message succeeds, then the protectedMessage element of the downlink Protected CPDLC Message shall be decoded as an ATCDownlinkMessage using unaligned basic PER and processed by the CPDLC-ground-user, as specified in 2.3.7.6 and 2.3.7.8

2.3.7.5.1.7 If the CPDLC Message received in a downlink Protected CPDLC Message 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].

2.3.7.5.2 Uplink Protected CPDLC Message

2.3.7.5.2.1 On receipt of an uplink Protected CPDLC Message, the Application Message Integrity Check verification procedure shall be performed before any other CPDLC Message processing.

2.3.7.5.2.2 On receipt of the first uplink Protected CPDLC Message, the CPDLC-air-user shall verify the Application Message Integrity Check verification procedure using:

a) the ATN Message Checksum verification procedure specified in 4.10.2., if the uplink Protected CPDLC Message does not contain an algorithm identifier or contains the algorithm identifier atn-default-checksum,

b) the specified algorithm if the uplink Protected CPDLC Message contains any other algorithm identifier.

2.3.7.5.2.3 On receipt of a subsequent uplink Protected CPDLC Message, the CPDLC-air-user shall :

a) ignore the specified algorithm if the uplink Protected CPDLC Message contains an algorithm identifier,

b) verify the Application Message Integrity Check using the Application Message Integrity Check verification procedure used for the preceding uplink Protected CPDLC Message.

2.3.7.5.2.4 On receipt of an uplink Protected CPDLC Message containing an unrecognised Application Message Integrity Check algorithm identifier, the CPDLC-air-user shall:

a) discard the received uplink Protected CPDLC Message, 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 Protected CPDLC Message that includes an unrecognised algorithm identifier.

2.3.7.5.2.5 When the Application Message Integrity Check verification procedure fails, the CPDLC-air-user shall:

a) discard the received uplink Protected CPDLC Message, 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 Protected CPDLC Message and the Application Message Integrity Check verification procedure fails.

2.3.7.5.2.6 If the Application Message Integrity Check verification procedure of an uplink Protected CPDLC Message succeeds, then the protectedMessage element of the uplink Protected CPDLC Message shall be decoded as an ATCUplinkMessage using unaligned basic PER and processed by the CPDLC-air-user as specified in 2.3.7.6 and 2.3.7.7.

2.3.7.5.2.7 If the CPDLC Message received in an uplink Protected CPDLC Message 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].

2.3.7.6 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 protectedMessage element of a Protected CPDLC Message. Before the CPDLC Message is processed as specified here, it will have successfully passed the Application Message Integrity Check verification procedure.

2.3.7.6.1 Message Attributes

Note 5.— Message attributes dictate certain message handling requirements for the CPDLC-user receiving a message. Each CPDLC Message has Urgency, Alert, and Response attributes.

Note 6.— Message element attribute table entries are listed in order of precedence (i.e., a precedence value of 1 is highest followed by 2, etc.).

2.3.7.6.1.1 When a message contains a single message element, the message attributes shall be the message element attributes.

2.3.7.6.1.2 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.

2.3.7.6.2 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.

2.3.7.6.2.1 Each message element shall have associated Urgency attributes with precedence as defined in table 2.3.7-1.

Table 2.3.7-1. Urgency Attribute (Air/Ground and Ground/Ground)

Type	Description	Precedence
D	Distress	1
U	Urgent	2
N	Normal	3
L	Low	4

2.3.7.6.2.2 When a CPDLC-user queues received messages, messages with the highest Urgency type shall be placed at the beginning of the queue.

2.3.7.6.2.3 When a CPDLC-user queues received messages, messages with the same Urgency type shall be queued in order of receipt.

2.3.7.6.3 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.

2.3.7.6.3.1 Each message element shall have associated Alert attributes with precedence as defined in table 2.3.7-2.

Table 2.3.7-2. Alert Attribute (Air/Ground and Ground/Ground)

Type	Description	Precedence
H	High	1
M	Medium	2
L	Low	3
N	No alerting required	4

2.3.7.6.3.2 Upon receipt of a CPDLC Message, the CPDLC-user shall provide one of three distinct alerts as determined by the received message alert attribute.

2.3.7.6.4 Response Attribute

Note.— The response (RESP) attribute mandates CPDLC-user response requirements for a given message element. Response message attribute only apply to air/ground messages.

2.3.7.6.4.1 Each uplink message element shall have associated Response attributes with precedence as defined in table 2.3.7-3.

Table 2.3.7-3. Response Attribute (Up-Link)

Type	*Response Required*	*Valid Responses Description*	*Precedence*
W/U	*Yes*	*Response required: WILCO, UNABLE, STANDBY, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY, ERROR, LOGICAL ACKNOWLEDGMENT (only if required)*	*1*
A/N	*Yes*	*Response required: AFFIRM, NEGATIVE, STANDBY, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY, ERROR, LOGICAL ACKNOWLEDGMENT (only if required)*	*2*
R	*Yes*	*Response required: ROGER, UNABLE, STANDBY, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY, ERROR, LOGICAL ACKNOWLEDGMENT (only if required),*	*3*
Y	*Yes*	*Any CPDLC downlink message, LOGICAL ACKNOWLEDGMENT (only if required),*	*4*

N	*No, unless logical acknowledgment required*	*LOGICAL ACKNOWLEDGMENT (only if required), ERROR, NOT CURRENT DATA AUTHORITY or NOT AUTHORIZED NEXT DATA AUTHORITY*	*5*

2.3.7.6.4.2 Each downlink message element shall have associated Response attributes with precedence as defined in table 2.3.7-4.

Table 2.3.7-4. Response Attribute (Down-Link)

Type	***Response Required***	***Valid Responses Description***	***Precedence***
Y	*Yes*	*Any CPDLC uplink message, LOGICAL ACKNOWLEDGMENT (only if required)*	*1*
N	*No, unless logical acknowledgment required*	*LOGICAL ACKNOWLEDGMENT (only if required), ERROR, SERVICE UNAVAILABLE, or FLIGHT PLAN NOT HELD*	*2*

2.3.7.6.5 CPDLC/DSC Distinction

2.3.7.6.5.1 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.

2.3.7.6.6 Air/Ground - Ground/Ground Distinction

2.3.7.6.6.1 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.

2.3.7.6.7 Logical Acknowledgment Prohibited

2.3.7.6.7.1 Upon receipt of the CPDLC Message USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED the CPDLC-air-user shall be prohibited from requiring a logical acknowledgment for any message sent for the duration of the CPDLC or DSC dialogue.

2.3.7.6.7.2 If the CPDLC-ground-user receives a CPDLC Message requiring a logical acknowledgment where the use of logical acknowledgment has been prohibited as above, the CPDLC-ground-user shall invoke the CPDLC-message service with a message containing the ERROR [errorinformation] message element with the [logicalAcknowledgmentNotAccepted] value as the *Protected CPDLC Message* parameter and discard the content of the received message.

2.3.7.6.8 Message Reference Numbers

2.3.7.6.8.1 If a received CPDLC Message requires a response, the CPDLC-user shall provide a message reference number for each response message sent.

2.3.7.6.8.2 The message reference number shall be identical to the message identification number of the received message to which it refers.

2.3.7.6.9 Message Response Requirements

2.3.7.6.9.1 **Recommendation.** — *A message sequence initiated by data link should be closed by data link.*

2.3.7.6.9.2 **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.*

2.3.7.6.9.3 A CPDLC-user shall only be permitted to respond to a received message in its entirety.

2.3.7.6.9.4 Only one closure response shall be permitted for a given message.

2.3.7.6.9.5 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 messages and then send a closure response message, or

b) send a closure response message.

2.3.7.6.9.6 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 messages and then send a closure response message, or

b) send a closure response message.

2.3.7.6.9.7 For a given message, once the CPDLC-user has sent the closure response message, no other response messages shall be sent referring to the given message.

2.3.7.6.9.8 When a message is received by the CPDLC-user requiring a logical acknowledgment 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 ACKNOWLEDGMENT 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 ACKNOWLEDGMENT message element, or with a message containing an ERROR, SERVICE UNAVAILABLE, or FLIGHT PLAN NOT HELD message element as appropriate.

2.3.7.6.9.9 A logical acknowledgment response message, if required, shall be sent prior to sending any other related response message(s), except:

a) when the response is an uplink message, a response message 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 message 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.

2.3.7.6.9.10 When the CPDLC-air-user receives a message with a W/U RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, WILCO, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

2.3.7.6.9.11 When the CPDLC-air-user receives a message with a W/U RESP attribute, the closure response message shall contain at least a WILCO, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

2.3.7.6.9.12 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 ACKNOWLEDGMENT (if required), STANDBY, AFFIRM, NEGATIVE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

2.3.7.6.9.13 When the CPDLC-air-user receives a message with an A/N RESP attribute, the closure response message shall contain at least a AFFIRM, NEGATIVE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

2.3.7.6.9.14 When the CPDLC-air-user receives a message with a R RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, ROGER, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

2.3.7.6.9.15 When the CPDLC-air-user receives a message with a R RESP attribute, the closure response message shall contain at least a ROGER, UNABLE, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element.

2.3.7.6.9.16 When the CPDLC-air-user receives a message with a Y RESP attribute, a LOGICAL ACKNOWLEDGMENT only when requested, and all other CPDLC Messages shall be permitted as a response message.

2.3.7.6.9.17 When the CPDLC-air-user receives a message with a Y RESP attribute, the first response message sent that does not contain a STANDBY or LOGICAL ACKNOWLEDGMENT shall constitute the closure response message.

2.3.7.6.9.18 When the CPDLC-ground-user receives an air/ground message with a Y RESP attribute, a LOGICAL ACKNOWLEDGMENT, only when requested and all other CPDLC Messages shall be permitted as a response message.

2.3.7.6.9.19 When the CPDLC-ground-user receives an air/ground message with a Y RESP attribute, the first response message sent that does not contain a STANDBY, REQUEST DEFERRED, or LOGICAL ACKNOWLEDGMENT message element shall constitute the closure response message.

2.3.7.6.9.20 When the CPDLC-air-user receives a message with a N RESP attribute, but requiring a logical acknowledgment, the only permitted response shall be a message that contains a LOGICAL ACKNOWLEDGMENT, NOT CURRENT DATA AUTHORITY, NOT AUTHORIZED NEXT DATA AUTHORITY or ERROR message element. This response message is the closure message.

2.3.7.6.9.21 When the CPDLC-ground-user receives an air/ground message with a N RESP attribute, but requiring a logical acknowledgment the only permitted response shall be a message that contains a LOGICAL ACKNOWLEDGMENT, SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR message element. This response message is the closure message.

2.3.7.6.9.22 When the CPDLC-ground-user receives a ground/ground message the ground-user shall be prohibited from generating a ground/ground response message.

Note.— Ground/ground forwarding of messages is a one-way exchange on a one message per dialogue basis. There are no message identification or message reference numbers contained in the header of a ground/ground message.

2.3.7.6.10 Invalid Message Elements

2.3.7.6.10.1 The CPDLC-ground-user shall be prohibited from sending any CPDLC Message containing the uplink message elements 33, 40, 41, or 178.

2.3.7.6.11 Error Conditions

2.3.7.6.11.1 Duplicate Message Identification Numbers

2.3.7.6.11.1.1 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.

2.3.7.6.11.2 Invalid Reference Number

2.3.7.6.11.2.1 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 as the *Protected CPDLC Message* parameter, and

b) disregard the received message.

2.3.7.6.11.3 No Available Message Identification Numbers

2.3.7.6.11.3.1 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.

2.3.7.6.11.4 Insufficient Resources

2.3.7.6.11.4.1 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 as the *Protected CPDLC Message* parameter, and

b) disregard the received message.

2.3.7.6.11.5 Invalid Message Element Combination

2.3.7.6.11.5.1 If a message is received containing:

a) a LOGICAL ACKNOWLEDGMENT message element in combination with any other message element in a single message,

b) a NEXT DATA AUTHORITY [facility] message element in combination with any other message element in a single message, or

c) a message containing more than two message elements with the [routeClearance] variable,

the CPDLC-user shall invoke the CPDLC-message request with the ERROR [errorinformation] message element with the [invalidMessageElementCombination] value as the *Protected CPDLC Message* parameter, and disregard the received message.

2.3.7.6.11.6 Invalid Message Elements

2.3.7.6.11.6.1 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 as the *Protected CPDLC Message* parameter, and disregard the received message.

2.3.7.6.11.7 System Management Responses

2.3.7.6.11.7.1 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 message, 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.

2.3.7.6.11.7.2 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 message, the SERVICE UNAVAILABLE, FLIGHT PLAN NOT HELD or ERROR message shall contain the received message identification number as the message reference number.

2.3.7.6.11.7.3 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.

2.3.7.6.11.7.4 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.

2.3.7.6.11.8 Invalid Message Response

2.3.7.6.11.8.1 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 ACKNOWLEDGMENT, 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].

2.3.7.6.11.8.2 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 ACKNOWLEDGMENT, 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].

2.3.7.6.11.8.3 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 ACKNOWLEDGMENT, 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].

2.3.7.6.11.8.4 If the CPDLC-ground-user sends a message that has a N response attribute and requires a logical acknowledgment, and a response to this message is received by the CPDLC-ground-user that does not contain any of the following message elements: LOGICAL ACKNOWLEDGMENT, 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].

2.3.7.6.11.8.5 If the CPDLC-air-user sends a message that has a N response attribute and requires a logical acknowledgment, and a response to this message is received by the CPDLC-air-user that does not contain any of the following message elements: LOGICAL ACKNOWLEDGMENT, 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].

2.3.7.6.11.9 Out of synchronisation

2.3.7.6.11.9.1 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.

2.3.7.6.11.10 Preventing Error loops

2.3.7.6.11.10.1 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.

2.3.7.6.11.10.2 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.

2.3.7.7 CPDLC-air-user Requirements

2.3.7.7.1 The CPDLC-start Service

2.3.7.7.1.1 Invoking the CPDLC-start request

2.3.7.7.1.1.1 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.

2.3.7.7.1.1.2 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.

2.3.7.7.1.1.3 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.

2.3.7.7.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response

2.3.7.7.1.2.1 Upon receipt of a CPDLC-start service indication, the CPDLC-air-user shall invoke a CPDLC-start service response within 0.5 seconds.

2.3.7.7.1.2.2 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 Message* parameter to a ProtectedDownlinkMessage containing a CPDLC Message element LOGICAL ACKNOWLEDGEMENT if a logical acknowlegment was requested and the received CPDLC message is not in error, or the CPDLC message 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 Message parameter to a ProtectedDownlinkMessage containing a CPDLC Message with the message element NOT AUTHORIZED NEXT DATA AUTHORITY.

2.3.7.7.1.2.3 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.

2.3.7.7.1.2.4 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 *Protected CPDLC Message* parameter shall be disregarded.

2.3.7.7.1.2.5 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 *Protected CPDLC Message* parameter shall be processed.

2.3.7.7.1.2.6 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 *Protected CPDLC Message* parameter shall be disregarded.

2.3.7.7.1.2.7 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.

2.3.7.7.1.2.8 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.

2.3.7.7.1.3 Receipt of a CPDLC-start confirmation

2.3.7.7.1.3.1 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.

2.3.7.7.1.3.2 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].

2.3.7.7.2 The DSC-start Service

2.3.7.7.2.1 Invoking the DSC-start request

2.3.7.7.2.1.1 Only a CPDLC-air-user shall be permitted to invoke the DSC-start service request primitive.

2.3.7.7.2.1.2 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.

2.3.7.7.2.1.3 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.

2.3.7.7.2.2 Receipt of a DSC-start confirmation

2.3.7.7.2.2.1 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.

2.3.7.7.3 The CPDLC-message Service

2.3.7.7.3.1 Invoking the CPDLC-message request

2.3.7.7.3.1.1 The CPDLC-air-user shall be prohibited from invoking the CPDLC-message service request primitive with a *Protected CPDLC Message* parameter not containing a CPDLC Message.

2.3.7.7.3.2 Receipt of a CPDLC-message Indication

2.3.7.7.3.2.1 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 CPDLC Message contained in the *Protected CPDLC Message* parameter.

2.3.7.7.3.2.2 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.

2.3.7.7.3.2.3 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.

2.3.7.7.3.2.4 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.

2.3.7.7.3.2.5 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 *Protected CPDLC Message* 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 *Protected CPDLC Message* parameter.

2.3.7.7.3.2.6 If a CPDLC-message indication is received containing a Protected CPDLC Message with no CPDLC Message message element, the CPDLC-air-user shall invoke CPDLC-user-abort request with the *Reason* parameter set to CPDLCUserAbortReason value [invalid-pdu].

2.3.7.7.4 The CPDLC-end Service

2.3.7.7.4.1 The CPDLC-end Service Request

2.3.7.7.4.1.1 The CPDLC-air-user shall be prohibited from invoking the CPDLC-end request.

2.3.7.7.4.2 Receipt of a CPDLC-end Indication and Invoking a CPDLC-end Response

2.3.7.7.4.2.1 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 *Protected CPDLC Message* 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 *Protected CPDLC Message* 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 2.3.7 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 Protected CPDLC Message parameter.

2.3.7.7.4.2.2 Uplink Message Contains Error

2.3.7.7.4.2.2.1 If a CPDLC-end indication is received from the Current Data Authority and there is a CPDLC Message in the *Protected CPDLC Message* parameter that either has the response attribute N and requires a logical acknowledgment or has a W/U, A/N, R, or Y response attribute, and an error is detected in the message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *Protected CPDLC Message* parameter containing a CPDLC Message with the message element ERROR [errorInformation],

b) the *Result* parameter set to the abstract value “rejected”.

2.3.7.7.4.2.3 No CPDLC Message in the CPDLC-end Indication

2.3.7.7.4.2.3.1 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 *Protected CPDLC Message* parameter, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value “accepted”.

2.3.7.7.4.2.3.2 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 *Protected CPDLC Message* parameter, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value “accepted” or “rejected”.

2.3.7.7.4.2.4 Message in CPDLC-end Indication Does Not Require Response

2.3.7.7.4.2.4.1 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 *Protected CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value “accepted”.

2.3.7.7.4.2.4.2 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 *Protected CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-air-user shall invoke CPDLC-end response with the *Result* parameter set to the abstract value “accepted” or “rejected”.

2.3.7.7.4.2.5 CPDLC Message in CPDLC-end Indication Requires Only Logical Acknowledgment

2.3.7.7.4.2.5.1 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user does not have any downlink open messages and there is a CPDLC message in the *Protected CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the CPDLC message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *Protected CPDLC Message* parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value “accepted”.

2.3.7.7.4.2.5.2 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has downlink open messages and there is a CPDLC Message in the *Protected CPDLC*

Message parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the CPDLC message, then the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *Protected CPDLC Message* parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.7.4.2.6 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.

2.3.7.7.4.2.6.1 If a CPDLC-end indication is received from the Current Data Authority and there is a CPDLC message in the *Protected CPDLC Message* parameter with the response attribute W/U, A/N or R and no error is detected in the message, then if the CPDLC-air-user chooses to respond positively (WILCO, AFFIRM, or ROGER) to the CPDLC message, then the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element WILCO, AFFIRM, or ROGER as appropriate, and

2) the *Result* parameter set to the abstract value "accepted".

2.3.7.7.4.2.7 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.

2.3.7.7.4.2.7.1 If a CPDLC-end indication is received from the Current Data Authority and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute W/U, A/N or R and no error is detected in the message, then if the CPDLC-air-user chooses to respond negatively (UNABLE or NEGATIVE) to the CPDLC Message, then the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element UNABLE or NEGATIVE as appropriate, and

2) the *Result* parameter set to the abstract value “rejected”.

2.3.7.7.4.2.8 CPDLC Message in CPDLC-end Indication With Y Response Attribute

2.3.7.7.4.2.8.1 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user does not have any downlink open messages and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute Y and no error is detected in the CPDLC Message, the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *Protected CPDLC Message* parameter containing a CPDLC Message with a Y attribute closure CPDLC Message, and

2) the *Result* parameter set to the abstract value “accepted”.

2.3.7.7.4.2.8.2 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has any downlink open messages and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute Y and no error is detected in the CPDLC Message, the CPDLC-air-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element STANDBY, and

c) invoke CPDLC-end response with:

1) the *Protected CPDLC Message* parameter containing CPDLC Message with a Y attribute closure CPDLC Message, and

2) the *Result* parameter set to the abstract value “accepted” or “rejected”.

2.3.7.7.4.2.9 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.

2.3.7.7.4.2.10 If the CPDLC-air-ASE associated with the Current Data Authority ceases to exist for any reason other than in response to a CPDLC-end request as specified above, any existing Next Data Authority designation and/or association shall cease to exist.

2.3.7.7.5 The DSC-end Service

2.3.7.7.5.1 The DSC-end Request

2.3.7.7.5.1.1 Only the CPDLC-air-user shall be permitted to invoke the DSC-end request.

2.3.7.7.5.1.2 If a CPDLC-air-user has invoked aDSC-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.

2.3.7.7.6 The CPDLC-user-abort Service

2.3.7.7.6.1 Issuing a CPDLC-user-abort Request [commanded-termination]

2.3.7.7.6.1.1 The CPDLC-air-user shall have the capability to invoke CPDLC-user-abort request with the *Reason* parameter set to CPDLCUserAbortReason value [commanded-termination].

2.3.7.7.6.2 Invoking a CPDLC-user-abort Request

2.3.7.7.6.2.1 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.

2.3.7.7.6.2.2 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.

2.3.7.7.6.3 Receipt of a CPDLC-abort Indication

2.3.7.7.6.3.1 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.

2.3.7.7.6.3.2 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.

2.3.7.8 CPDLC-ground-user Requirements

2.3.7.8.1 The CPDLC-start Service

2.3.7.8.1.1 Invoking the CPDLC-start request

2.3.7.8.1.1.1 If there is no CPDLC-service, the only CPDLC-service primitives the CPDLC-ground-user shall be permitted to invoke are the CPDLC-start-request or the CPDLC-forward request.

2.3.7.8.1.1.2 If a CPDLC-ground-user has invoked a CPDLC-start request, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive on the CPDLC dialogue intiated by the CPDLC-start service request, except the CPDLC-user-abort request until after it has received a CPDLC-start confirmation.

2.3.7.8.1.2 Receipt of a CPDLC-start Indication and Invoking CPDLC-start Response

2.3.7.8.1.2.1 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.

2.3.7.8.1.2.2 The CPDLC-ground-user shall be prohibited from invoking the CPDLC-start response unless and until it has received a CPDLC-start indication.

2.3.7.8.1.2.3 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value "rejected" then the *Response Message* parameter shall be a ProtectedUplinkMessage containing a CPDLC Message with either the SERVICE UNAVAILABLE or FLIGHT PLAN NOT HELD message element as appropriate.

2.3.7.8.1.2.4 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 *Protected CPDLC Message* parameter.

2.3.7.8.1.2.5 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 *Protected CPDLC Message* parameter shall be processed.

2.3.7.8.1.2.6 When the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value "accepted", it shall set the *Response Message* parameter to a ProtectedUplinkMessage containing:

a) A CPDLC Message with the message element LOGICAL AKNOWLEDGEMENT if a logical aknowledgement 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.

2.3.7.8.1.2.7 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 a 24 bit aircraft address contained in CPDLC-start indication *Calling Peer Identifier* parameter.

2.3.7.8.1.2.8 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.

2.3.7.8.1.2.9 Upon receipt of a CPDLC-start service indication, the CPDLC-ground-user shall invoke a CPDLC-start service response within 0.5 seconds.

2.3.7.8.1.3 Receipt of a CPDLC-start confirmation

2.3.7.8.1.3.1 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 a 24 bit aircraft address contained in CPDLC-start request *Called Peer Identifier* parameter.

2.3.7.8.1.3.2 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].

2.3.7.8.2 The DSC-start Service

2.3.7.8.2.1 Receipt of a DSC-start Indication and Invoking DSC-start Response

2.3.7.8.2.1.1 The CPDLC-ground-user shall be prohibited from invoking the DSC-start response unless and until it has received a DSC-start indication.

2.3.7.8.2.1.2 If a DSC-start indication is received from an aircraft with which the ground system currently has a DSC dialogue, the CPDLC-ground-user shall:

a) invoke the DSC-start response with the *Result* parameter set to the abstract value “accepted”, and

b) invoke the CPDLC-user-abort request for the first DSC dialogue with that aircraft.

2.3.7.8.2.1.3 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "rejected" then the *Response Message* parameter a ProtectedUplinkMessage containing a CPDLC Message with either the SERVICE UNAVAILABLE or FLIGHT PLAN NOT HELD message element as appropriate.

2.3.7.8.2.1.4 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "rejected" the CPDLC-ground-user shall disregard any CPDLC Message contained in the DSC-start indication *Protected CPDLC Message* parameter.

2.3.7.8.2.1.5 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "accepted" any CPDLC Message contained in the DSC-start indication *Protected CPDLC Message* parameter shall be processed.

2.3.7.8.2.1.6 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "accepted" the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in the DSC-start indication *Aircraft Address* parameter.

2.3.7.8.2.1.7 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.

2.3.7.8.2.2 Receipt of a DSC-start Indication and Invoking a DSC-start Response

2.3.7.8.2.2.1 Upon receipt of a DSC-start indication, the CPDLC-ground-user shall invoke a DSC-start response within 0.5 seconds.

2.3.7.8.2.2.2 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 Message* parameter to a ProtectedUplinkMessage containing a CPDLC Message with the message element SERVICE UNAVAILABLE or FLIGHT PLAN NOT HELD as appropriate.

2.3.7.8.3 The CPDLC-message Service

2.3.7.8.3.1 Invoking the CPDLC-message request

2.3.7.8.3.1.1 The CPDLC-ground-user shall be prohibited from invoking the CPDLC-message service request primitive with a *Protected CPDLC Message* parameter not containing a CPDLC Message.

2.3.7.8.3.2 Receipt of a CPDLC-message indication

2.3.7.8.3.2.1 If a CPDLC-message indication is received containing a Protected CPDLC Message with no CPDLC Message message element, the CPDLC-ground-user shall invoke CPDLC-user-abort request with the *Reason* parameter set to CPDLCUserAbortReason value [invalid-pdu].

2.3.7.8.4 The CPDLC-end Service

2.3.7.8.4.1 The CPDLC-end Request

2.3.7.8.4.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-end request.

2.3.7.8.4.1.2 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.

2.3.7.8.5 The DSC-end Service

2.3.7.8.5.1 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 2.3.7 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 Protected CPDLC Message parameter.

2.3.7.8.5.1.1 Downlink Message Contains Error

2.3.7.8.5.1.1.1 If a DSC-end indication is received and there is a CPDLC Message in the *Protected CPDLC Message* parameter that either has the response attribute N and requires a logical acknowledgment or has Y response attribute, and an error is detected in the CPDLC Message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the *Protected CPDLC Message* parameter containing CPDLC Message with the message element ERROR [errorInformation],

b) the *Result* parameter set to the abstract value "rejected".

2.3.7.8.5.1.2 No CPDLC Message in the DSC-end Indication

2.3.7.8.5.1.2.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is no CPDLC Message in the *Protected CPDLC Message* parameter, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted".

2.3.7.8.5.1.2.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is no CPDLC Message in the *Protected CPDLC Message* parameter, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.8.5.1.3 CPDLC Message in DSC-end Indication Does Not Require Response

2.3.7.8.5.1.3.1 If a DSC-end indication is received and the CPDLC-ground-user does not have any uplink open messages and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-ground-user shall invoke DSC-end response with the *Result* parameter set to the abstract value "accepted".

2.3.7.8.5.1.3.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute N and not requiring a logical acknowledgment, then the CPDLC-ground-user shall invokeDSC-end response with the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.8.5.1.4 CPDLC Message inDSC-end Indication Requires Only Logical Acknowledgment

2.3.7.8.5.1.4.1 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 *Protected CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the CPDLC Message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the *Protected CPDLC Message* parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value "accepted".

2.3.7.8.5.1.4.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute N and requiring a logical acknowledgment, and no error is detected in the CPDLC Message, then the CPDLC-ground-user shall invoke DSC-end response with:

a) the *Protected CPDLC Message* parameter containing a CPDLC Message with the message element LOGICAL ACKNOWLEDGMENT, and

b) the *Result* parameter set to the abstract value "accepted" or "rejected".

2.3.7.8.5.1.5 CPDLC Message in DSC-end Indication With Y Response Attribute

2.3.7.8.5.1.5.1 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 *Protected CPDLC Message* parameter with the response attribute Y and no error is detected in the CPDLC Message the CPDLC-ground-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element STANDBY,

c) if a REQUEST DEFERRED response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element REQUEST DEFERRED, and

d) invoke DSC-end response with:

1) the *Protected CPDLC Message* parameter containing a CPDLC Message with a Y attribute closure CPDLC Message, and

2) the *Result* parameter set to the abstract value "accepted".

2.3.7.8.5.1.5.2 If a DSC-end indication is received and the CPDLC-ground-user has uplink open messages and there is a CPDLC Message in the *Protected CPDLC Message* parameter with the response attribute Y and no error is detected in the message the CPDLC-ground-user shall:

a) if a logical acknowledgment is required, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with only the message element LOGICAL ACKNOWLEDGMENT,

b) if a STANDBY response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element STANDBY, and

c) if a REQUEST DEFERRED response is used, invoke CPDLC-message request with the *Protected CPDLC Message* parameter containing a CPDLC Message with at least the message element REQUEST DEFERRED, and

d) invoke DSC-end response with:

1) the *Protected CPDLC Message* 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".

2.3.7.8.6 The CPDLC-forward Service

2.3.7.8.6.1 Invoking the CPDLC-forward Request

2.3.7.8.6.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-forward Request.

2.3.7.8.6.1.2 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.

2.3.7.8.7 The CPDLC-user-abort Service

2.3.7.8.7.1 Issuing a CPDLC-user-abort Request

2.3.7.8.7.1.1 The CPDLC-ground-user shall have the capability to invoke CPDLC-user-abort request with the *Reason* parameter set to CPDLCUserAbortReason value [commanded-termination].

2.3.7.9 Message Intent

2.3.7.9.1 Purpose

Note 1.— 2.3.7.9 contains the message set for CPDLC Message attributes, message presentation guidance, and data structure presentation guidance are presented. The actual information exchanged between an aircraft and ground peer or a ground and ground peer CPDLC applications is defined in 2.3.4; however, 2.3.4 does not mandate any particular method for presenting this information. The presentation of information to the controller and aircraft crew is a local implementation. The message presentation recommendations contained in Tables 2.3.7-5 to 2.3.7-28 are one possible means of presenting the information. These recommendations are generally consistent with current ICAO practices for displaying ATC information.

Note 2.— Tables 2.3.7-5 to 2.3.7-28 which are aligned with the provisions contained in Appendix 5 of the 13th Edition (including Amendment 2, dated 5 November 1998) of ICAO Doc 4444 — Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services (PANS-RAC) *are included in this document for completeness.*

2.3.7.9.2 Uplink message elements shall comply with the intent, use, and element attributes as presented in Tables 2.3.7-5 to 2.3.7-16.

Table 2.3.7-5. Responses/Acknowledgments (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
0	Indicates that ATC cannot comply with the request.	UNABLE	N	M	N
1	Indicates that ATC has received the message and will respond.	STANDBY	N	L	N
2	Indicates that ATC has received the request but it has been deferred until later.	REQUEST DEFERRED	N	L	N
3	Indicates that ATC has received and understood the message.	ROGER	N	L	N
4	Yes.	AFFIRM	N	L	N
5	No.	NEGATIVE	N	L	N
235	Notification of receipt of unlawful interference message.	ROGER 7500	U	H	N
211	Indicates that the ATC has received the request and has passed it to the next control authority.	REQUEST FORWARDED	N	L	N
218	Indicates to the pilot that the request has already been received on the ground.	REQUEST ALREADY RECEIVED	L	N	N
237	Indicates that the request cannot be responded to by the current unit, and that it should be requested from the next unit	REQUEST AGAIN WITH NEXT UNIT	N	L	N

Table 2.3.7-6. Vertical Clearances (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Notification that a level change instruction should be expected.	EXPECT [level]	L	L	R
7	Notification that an instruction should be expected for the aircraft to commence climb at the specified time.	EXPECT CLIMB AT [time]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
8	Notification that an instruction should be expected for the aircraft to commence climb at the specified position.	EXPECT CLIMB AT [position]	L	L	R
9	Notification that an instruction should be expected for the aircraft to commence descent at the specified time.	EXPECT DESCENT AT [time]	L	L	R
10	Notification that an instruction should be expected for the aircraft to commence descent at the specified position.	EXPECT DESCENT AT [position]	L	L	R
11	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified time.	EXPECT CRUISE CLIMB AT [time]	L	L	R
12	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified position.	EXPECT CRUISE CLIMB AT [position]	L	L	R
13	Notification that an instruction should be expected for the aircraft to commence climb at the specified time to the specified level.	AT [time] EXPECT CLIMB TO [level]	L	L	R
14	Notification that an instruction should be expected for the aircraft to commence climb at the specified position to the specified level.	AT [position] EXPECT CLIMB TO [level]	L	L	R
15	Notification that an instruction should be expected for the aircraft to commence descent at the specified time to the specified level.	AT [time] EXPECT DESCENT TO [level]	L	L	R

	Message Intent/Use	Message Element	URG	ALRT	RESP
16	Notification that an instruction should be expected for the aircraft to commence descent at the specified position to the specified level.	AT [position] EXPECT DESCENT TO [level]	L	L	R
17	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified time to the specified level.	AT [time] EXPECT CRUISE CLIMB TO [level]	L	L	R
18	Notification that an instruction should be expected for the aircraft to commence cruise climb at the specified position to the specified level.	AT [position] EXPECT CRUISE CLIMB TO [level]	L	L	R
19	Instruction to maintain the specified level.	MAINTAIN [level]	N	M	W/U
20	Instruction that a climb to a specified level is to commence and once reached the specified level is to be maintained.	CLIMB TO [level]	N	M	W/U
21	Instruction that at the specified time a climb to the specified level is to commence and once reached the specified level is to be maintained.	AT [time] CLIMB TO [level]	N	M	W/U
22	Instruction that at the specified position a climb to the specified level is to commence and once reached the specified level is to be maintained.	AT [position] CLIMB TO [level]	N	M	W/U
185	Instruction that after passing the specified position a climb to the specified level is to commence and once reached the specified level is to be maintained.	AFTER PASSING [position] CLIMB TO [level]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
23	Instruction that a descent to a specified level is to commence and once reached the specified level is to be maintained.	DESCEND TO [level]	N	M	W/U
24	Instruction that at a specified time a descent to a specified level is to commence and once reached the specified level is to be maintained.	AT [time] DESCEND TO [level]	N	M	W/U
25	Instruction that at the specified position a descent to the specified level is to commence and once reached the specified level is to be maintained.	AT [position] DESCEND TO [level]	N	M	W/U
186	Instruction that after passing the specified position a descent to the specified level is to commence and once reached the specified level is to be maintained.	AFTER PASSING [position] DESCEND TO [level]	N	M	W/U
26	Instruction that a climb is to commence at a rate such that the specified level is reached at or before the specified time.	CLIMB TO REACH [level] BY [time]	N	M	W/U
27	Instruction that a climb is to commence at a rate such that the specified level is reached at or before the specified position.	CLIMB TO REACH [level] BY [position]	N	M	W/U
28	Instruction that a descent is to commence at a rate such that the specified level is reached at or before the specified time.	DESCEND TO REACH [level] BY [time]	N	M	W/U
29	Instruction that a descent is to commence at a rate such that the specified level is reached at or before the specified position.	DESCEND TO REACH [level] BY [position]	N	M	W/U
192	Instruction that a change of level is to continue, but at a rate such that the specified level is reached at or before the specified time.	REACH [level] BY [time]	N	M	W/U

	Message Intent/Use	Message Element	URG	ALRT	RESP
209	Instruction that a change of level is to continue, but at a rate such that the specified level is reached at or before the specified position.	REACH [level] BY [position]	N	M	W/U
30	Instruction that a level within the defined vertical range specified is to be maintained.	MAINTAIN BLOCK [level] TO [level]	N	M	W/U
31	Instruction that a climb to a level within the vertical range defined is to commence.	CLIMB TO AND MAINTAIN BLOCK [level] TO [level]	N	M	W/U
32	Instruction that a descent to a level within the vertical range defined is to commence.	DESCEND TO AND MAINTAIN BLOCK [level] TO [level]	N	M	W/U
34	Instruction that a cruise climb is to commence and continue until the specified level is reached.	CRUISE CLIMB TO [level]	N	M	W/U
35	Instruction that a cruise climb can commence once above the specified level.	CRUISE CLIMB ABOVE [level]	N	M	W/U
219	Instruction to stop the climb below the previously assigned level.	STOP CLIMB AT [level]	U	M	W/U
220	Instruction to stop the descent above the previously assigned level.	STOP DESCENT AT [level]	U	M	W/U
36	Instruction that the climb to the specified level should be made at the aircraft's best rate.	EXPEDITE CLIMB TO [level]	U	M	W/U
37	Instruction that the descent to the specified level should be made at the aircraft's best rate.	EXPEDITE DESCENT TO [level]	U	M	W/U
38	Urgent instruction to immediately climb to the specified level.	IMMEDIATELY CLIMB TO [level]	D	H	W/U
39	Urgent instruction to immediately descend to the specified level.	IMMEDIATELY DESCEND TO [level]	D	H	W/U

	Message Intent/Use	Message Element	URG	ALRT	RESP
40	(reserved)		L	L	Y
41	(reserved)		L	L	Y
171	Instruction to climb at not less than the specified rate.	CLIMB AT [vertical rate] MINIMUM	N	M	W/U
172	Instruction to climb at not above the specified rate.	CLIMB AT [vertical rate] MAXIMUM	N	M	W/U
173	Instruction to descend at not less than the specified rate.	DESCEND AT [vertical rate] MINIMUM	N	M	W/U
174	Instruction to descend at not above the specified rate.	DESCEND AT [vertical rate] MAXIMUM	N	M	W/U
33	(reserved)		L	L	Y

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-7. Crossing Constraints (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
42	Notification that a level change instruction should be expected which will require the specified position to be crossed at the specified level.	EXPECT TO CROSS [position] AT [level]	L	L	R
43	Notification that a level change instruction should be expected which will require the specified position to be crossed at or above the specified level.	EXPECT TO CROSS [position] AT OR ABOVE [level]	L	L	R
44	Notification that a level change instruction should be expected which will require the specified position to be crossed at or below the specified level.	EXPECT TO CROSS [position] AT OR BELOW [level]	L	L	R

	Message Intent/Use	Message Element	URG	ALRT	RESP
45	Notification that a level change instruction should be expected which will require the specified position to be crossed at the specified level which is to be maintained subsequently.	EXPECT TO CROSS [position] AT AND MAINTAIN [level]	L	L	R
46	Instruction that the specified position is to be crossed at the specified level. This may require the aircraft to modify its climb or descent profile.	CROSS [position] AT [level]	N	M	W/U
47	Instruction that the specified position is to be crossed at or above the specified level.	CROSS [position] AT OR ABOVE [level]	N	M	W/U
48	Instruction that the specified position is to be crossed at or below the specified level.	CROSS [position] AT OR BELOW [level]	N	M	W/U
49	Instruction that the specified position is to be crossed at the specified level and that level is to be maintained when reached.	CROSS [position] AT AND MAINTAIN [level]	N	M	W/U
50	Instruction that the specified position is to be crossed at a level between the specified levels.	CROSS [position] BETWEEN [level] AND [level]	N	M	W/U
51	Instruction that the specified position is to be crossed at the specified time.	CROSS [position] AT [time]	N	M	W/U
52	Instruction that the specified position is to be crossed at or before the specified time.	CROSS [position] AT OR BEFORE [time]	N	M	W/U
53	Instruction that the specified position is to be crossed at or after the specified time.	CROSS [position] AT OR AFTER [time]	N	M	W/U
54	Instruction that the specified position is to be crossed at a time between the specified times.	CROSS [position] BETWEEN [time] AND [time]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
55	Instruction that the specified position is to be crossed at the specified speed and the specified speed is to be maintained until further advised.	CROSS [position] AT [speed]	N	M	W/U
56	Instruction that the specified position is to be crossed at a speed equal to or less than the specified speed and the specified speed or less is to be maintained until further advised.	CROSS [position] AT OR LESS THAN [speed]	N	M	W/U
57	Instruction that the specified position is to be crossed at a speed equal to or greater than the specified speed and the specified speed or greater is to be maintained until further advised.	CROSS [position] AT OR GREATER THAN [speed]	N	M	W/U
58	Instruction that the specified position is to be crossed at the specified time and the specified level.	CROSS [position] AT [time] AT [level]	N	M	W/U
59	Instruction that the specified position is to be crossed at or before the specified time and at the specified level.	CROSS [position] AT OR BEFORE [time] AT [level]	N	M	W/U
60	Instruction that the specified position is to be crossed at or after the specified time and at the specified level.	CROSS [position] AT OR AFTER [time] AT [level]	N	M	W/U
61	Instruction that the specified position is to be crossed at the specified level and speed, and the level and speed are to be maintained.	CROSS [position] AT AND MAINTAIN [level] AT [speed]	N	M	W/U
62	Instruction that at the specified time the specified position is to be crossed at the specified level and the level is to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [level]	N	M	W/U

	Message Intent/Use	Message Element	URG	ALRT	RESP
63	Instruction that at the specified time the specified position is to be crossed at the specified level and speed, and the level and speed are to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [level] AT [speed]	N	M	W/U

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-8. Lateral Offsets (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
64	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction.	OFFSET [specified distance] [direction] OF ROUTE	N	M	W/U
65	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction and commencing at the specified position.	AT [position] OFFSET [specified distance] [direction] OF ROUTE	N	M	W/U
66	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction and commencing at the specified time.	AT [time] OFFSET [specified distance] [direction] OF ROUTE	N	M	W/U
67	Instruction that the cleared flight route is to be rejoined.	PROCEED BACK ON ROUTE	N	M	W/U
68	Instruction that the cleared flight route is to be rejoined at or before the specified position.	REJOIN ROUTE BY [position]	N	M	W/U
69	Instruction that the cleared flight route is to be rejoined at or before the specified time.	REJOIN ROUTE BY [time]	N	M	W/U
70	Notification that a clearance may be issued to enable the aircraft to rejoin the cleared route at or before the specified position.	EXPECT BACK ON ROUTE BY [position]	L	L	R

	Message Intent/Use	Message Element	URG	ALRT	RESP
71	Notification that a clearance may be issued to enable the aircraft to rejoin the cleared route at or before the specified time.	EXPECT BACK ON ROUTE BY [time]	L	L	R
72	Instruction to resume own navigation following a period of tracking or heading clearances. May be used in conjunction with an instruction on how or where to rejoin the cleared route.	RESUME OWN NAVIGATION	N	M	W/U

Table 2.3.7-9. Route Modifications (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
73	Notification to the aircraft of the instructions to be followed from departure until the specified clearance limit.	[departure clearance]	N	M	W/U
74	Instruction to proceed directly from its present position to the specified position.	PROCEED DIRECT TO [position]	N	M	W/U
75	Instruction to proceed, when able, directly to the specified position.	WHEN ABLE PROCEED DIRECT TO [position]	N	M	W/U
76	Instruction to proceed, at the specified time, directly to the specified position.	AT [time] PROCEED DIRECT TO [position]	N	M	W/U
77	Instruction to proceed, at the specified position, directly to the next specified position.	AT [position] PROCEED DIRECT TO [position]	N	M	W/U
78	Instruction to proceed, upon reaching the specified level, directly to the specified position.	AT [level] PROCEED DIRECT TO [position]	N	M	W/U
79	Instruction to proceed to the specified position via the specified route.	CLEARED TO [position] VIA [route clearance]	N	M	W/U
80	Instruction to proceed via the specified route.	CLEARED [route clearance]	N	M	W/U

	Message Intent/Use	Message Element	URG	ALRT	RESP
81	Instruction to proceed in accordance with the specified procedure.	CLEARED [procedure name]	N	M	W/U
236	Instruction to leave controlled airspace.	LEAVE CONTROLLED AIRSPACE	N	M	W/U
82	Approval to deviate up to the specified distance from the cleared route in the specified direction.	CLEARED TO DEVIATE UP TO [specified distance] [direction] OF ROUTE	N	M	W/U
83	Instruction to proceed from the specified position via the specified route.	AT [position] CLEARED [route clearance]	N	M	W/U
84	Instruction to proceed from the specified position via the specified procedure.	AT [position] CLEARED [procedure name]	N	M	W/U
85	Notification that a clearance to fly on the specified route may be issued.	EXPECT [route clearance]	L	L	R
86	Notification that a clearance to fly on the specified route from the specified position may be issued.	AT [position] EXPECT [route clearance]	L	L	R
87	Notification that a clearance to fly directly to the specified position may be issued.	EXPECT DIRECT TO [position]	L	L	R
88	Notification that a clearance to fly directly from the first specified position to the next specified position may be issued.	AT [position] EXPECT DIRECT TO [position]	L	L	R
89	Notification that a clearance to fly directly to the specified position commencing at the specified time may be issued.	AT [time] EXPECT DIRECT TO [position]	L	L	R
90	Notification that a clearance to fly directly to the specified position commencing when the specified level is reached may be issued.	AT [level] EXPECT DIRECT TO [position]	L	L	R

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
91	Instruction to enter a holding pattern with the specified characteristics at the specified position and level.	HOLD AT [position] MAINTAIN [level] INBOUND TRACK [degrees] [direction] TURNS [leg type]	N	M	W/U
92	Instruction to enter a holding pattern with the published characteristics at the specified position and level.	HOLD AT [position] AS PUBLISHED MAINTAIN [level]	N	M	W/U
93	Notification that an onwards clearance may be issued at the specified time.	EXPECT FURTHER CLEARANCE AT [time]	L	L	R
94	Instruction to turn left or right as specified on to the specified heading.	TURN [direction] HEADING [degrees]	N	M	W/U
95	Instruction to turn left or right as specified on to the specified track.	TURN [direction] GROUND TRACK [degrees]	N	M	W/U
215	Instruction to turn a specified number of degrees left or right.	TURN [direction] [degrees]	N	M	W/U
190	Instruction to fly on the specified heading.	FLY HEADING [degrees]	N	M	W/U
96	Instruction to continue to fly on the current heading.	CONTINUE PRESENT HEADING	N	M	W/U
97	Instruction to fly on the specified heading from the specified position.	AT [position] FLY HEADING [degrees]	N	M	W/U
221	Instruction to stop turn at the specified heading prior to reaching the previously assigned heading.	STOP TURN HEADING [degrees]	U	M	W/U
98	Instruction to turn immediately left or right as specified on to the specified heading.	IMMEDIATELY TURN [direction] HEADING [degrees]	D	H	W/U
99	Notification that a clearance may be issued for the aircraft to fly the specified procedure.	EXPECT [procedure name]	L	L	R

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-10. Speed Changes (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
100	Notification that a speed instruction may be issued to be effective at the specified time.	AT [time] EXPECT [speed]	L	L	R
101	Notification that a speed instruction may be issued to be effective at the specified position.	AT [position] EXPECT [speed]	L	L	R
102	Notification that a speed instruction may be issued to be effective at the specified level.	AT [level] EXPECT [speed]	L	L	R
103	Notification that a speed range instruction may be issued to be effective at the specified time.	AT [time] EXPECT [speed] TO [speed]	L	L	R
104	Notification that a speed range instruction may be issued to be effective at the specified position.	AT [position] EXPECT [speed] TO [speed]	L	L	R
105	Notification that a speed range instruction may be issued to be effective at the specified level.	AT [level] EXPECT [speed] TO [speed]	L	L	R
106	Instruction that the specified speed is to be maintained.	MAINTAIN [speed]	N	M	W/U
188	Instruction that after passing the specified position the specified speed is to be maintained.	AFTER PASSING [position] MAINTAIN [speed]	N	M	W/U
107	Instruction that the present speed is to be maintained.	MAINTAIN PRESENT SPEED	N	M	W/U
108	Instruction that the specified speed or a greater speed is to be maintained.	MAINTAIN [speed] OR GREATER	N	M	W/U
109	Instruction that the specified speed or a lesser speed is to be maintained.	MAINTAIN [speed] OR LESS	N	M	W/U
110	Instruction that a speed within the specified range is to be maintained.	MAINTAIN [speed] TO [speed]	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
111	Instruction that the present speed is to be increased to the specified speed and maintained until further advised.	INCREASE SPEED TO [speed]	N	M	W/U
112	Instruction that the present speed is to be increased to the specified speed or greater, and maintained at or above the specified speed until further advised.	INCREASE SPEED TO [speed] OR GREATER	N	M	W/U
113	Instruction that the present speed is to be reduced to the specified speed and maintained until further advised.	REDUCE SPEED TO [speed]	N	M	W/U
114	Instruction that the present speed is to be reduced to the specified speed or less and maintained at or below the specified speed until further advised.	REDUCE SPEED TO [speed] OR LESS	N	M	W/U
115	Instruction that the specified speed is not to be exceeded.	DO NOT EXCEED [speed]	N	M	W/U
116	Notification that the aircraft need no longer comply with the previously issued speed restriction.	RESUME NORMAL SPEED	N	M	W/U
189	Instruction that the present speed is to be changed to the specified speed.	ADJUST SPEED TO [speed]	N	M	W/U
222	Notification that the aircraft may keep its preferred speed without restriction.	NO SPEED RESTRICTION	L	L	R
223	Instruction to reduce present speed to the minimum safe approach speed.	REDUCE TO MINIMUM APPROACH SPEED	N	M	W/U

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-11. Contact/Monitor/Surveillance Requests (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
117	Instruction that the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	CONTACT [unitname] [frequency]	N	M	W/U
118	Instruction that at the specified position the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [position] CONTACT [unitname] [frequency]	N	M	W/U
119	Instruction that at the specified time the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [time] CONTACT [unitname] [frequency]	N	M	W/U
120	Instruction that the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	MONITOR [unitname] [frequency]	N	M	W/U
121	Instruction that at the specified position the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [position] MONITOR [unitname] [frequency]	N	M	W/U
122	Instruction that at the specified time the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [time] MONITOR [unitname] [frequency]	N	M	W/U
123	Instruction that the specified code (SSR code) is to be selected.	SQUAWK [code]	N	M	W/U
124	Instruction that the SSR transponder responses are to be disabled.	STOP SQUAWK	N	M	W/U
125	Instruction that the SSR transponder responses should include level information.	SQUAWK MODE CHARLIE	N	M	W/U

	Message Intent/Use	Message Element	URG	ALRT	RESP
126	Instruction that the SSR transponder responses should no longer include level information.	STOP SQUAWK MODE CHARLIE	N	M	W/U
179	Instruction that the 'ident' function on the SSR transponder is to be actuated.	SQUAWK IDENT	N	M	W/U

Table 2.3.7-12. Report/Confirmation Requests (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
127	Instruction to report when the aircraft is back on the cleared route.	REPORT BACK ON ROUTE	N	L	W/U
128	Instruction to report when the aircraft has left the specified level.	REPORT LEAVING [level]	N	L	W/U
129	Instruction to report when the aircraft is in level flight at the specified level.	REPORT MAINTAINING [level]	N	L	W/U
175	Instruction to report when the aircraft has reached the specified level.	REPORT REACHING [level]	N	L	W/U
200	Instruction used in conjunction with a level clearance to report reaching the level assigned.	REPORT REACHING	N	M	W/U
180	Instruction to report when the aircraft is within the specified vertical range.	REPORT REACHING BLOCK [level] TO [level]	N	L	W/U
130	Instruction to report when the aircraft has passed the specified position.	REPORT PASSING [position]	N	L	W/U
181	Instruction to report the present distance to or from the specified position.	REPORT DISTANCE [to/from] [position]	N	M	Y
184	Instruction to report at the specified time the distance to or from the specified position.	AT TIME [time] REPORT DISTANCE [to/from] [position]	N	L	Y

	Message Intent/Use	Message Element	URG	ALRT	RESP
228	Instruction to report the estimated time of arrival at the specified position.	REPORT ETA [position]	L	L	Y
131	Instruction to report the amount of fuel remaining and the number of persons on board.	REPORT REMAINING FUEL AND PERSONS ON BOARD	U	M	Y
132	Instruction to report the present position.	REPORT POSITION	N	M	Y
133	Instruction to report the present level.	REPORT PRESENT LEVEL	N	M	Y
134	Instruction to report the requested speed.	REPORT [speed type] [speed type] [speed type] SPEED	N	M	Y
135	Instruction to confirm and acknowledge the currently assigned level.	CONFIRM ASSIGNED LEVEL	N	L	Y
136	Instruction to confirm and acknowledge the currently assigned speed.	CONFIRM ASSIGNED SPEED	N	L	Y
137	Instruction to confirm and acknowledge the currently assigned route.	CONFIRM ASSIGNED ROUTE	N	L	Y
138	Instruction to confirm the previously reported time over the last reported waypoint.	CONFIRM TIME OVER REPORTED WAYPOINT	N	L	Y
139	Instruction to confirm the identity of the previously reported waypoint.	CONFIRM REPORTED WAYPOINT	N	L	Y
140	Instruction to confirm the identity of the next waypoint.	CONFIRM NEXT WAYPOINT	N	L	Y
141	Instruction to confirm the previously reported estimated time at the next waypoint.	CONFIRM NEXT WAYPOINT ETA	N	L	Y
142	Instruction to confirm the identity of the next but one waypoint.	CONFIRM ENSUING WAYPOINT	N	L	Y
143	The request was not understood. It should be clarified and resubmitted.	CONFIRM REQUEST	N	L	Y

	Message Intent/Use	Message Element	URG	ALRT	RESP
144	Instruction to report the selected (SSR) code.	CONFIRM SQUAWK	N	L	Y
145	Instruction to report the present heading.	REPORT HEADING	N	M	Y
146	Instruction to report the present ground track.	REPORT GROUND TRACK	N	M	Y
182	Instruction to report the identification code of the last ATIS received.	CONFIRM ATIS CODE	N	L	Y
147	Instruction to make a position report.	REQUEST POSITION REPORT	N	M	Y
216	Instruction to file a flight plan.	REQUEST FLIGHT PLAN	N	M	Y
217	Instruction to report that the aircraft has landed.	REPORT ARRIVAL	N	M	Y
229	Instruction to report the preferred alternate aerodrome for landing.	REPORT ALTERNATE AERODROME	L	L	Y
231	Instruction to indicate the pilot's preferred level.	STATE PREFERRED LEVEL	L	L	Y
232	Instruction to indicate the pilot's preferred time and/or position to commence descent to the aerodrome of intended arrival.	STATE TOP OF DESCENT	L	L	Y

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-13. Negotiation Requests (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
148	Request for the earliest time at which the specified level can be accepted.	WHEN CAN YOU ACCEPT [level]	N	L	Y
149	Instruction to report whether or not the specified level can be accepted at the specified position.	CAN YOU ACCEPT [level] AT [position]	N	L	A/N
150	Instruction to report whether or not the specified level can be accepted at the specified time.	CAN YOU ACCEPT [level] AT [time]	N	L	A/N

151	Instruction to report the earliest time when the specified speed can be accepted.	WHEN CAN YOU ACCEPT [speed]	N	L	Y
152	Instruction to report the earliest time when the specified offset track can be accepted.	WHEN CAN YOU ACCEPT [specified distance] [direction] OFFSET	N	L	Y

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-14. Air Traffic Advisories (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
153	ATS advisory that the altimeter setting should be the specified setting.	ALTIMETER [altimeter]	N	L	R
213	ATS advisory that the specified altimeter setting relates to the specified facility.	[facility designation] ALTIMETER [altimeter]	N	L	R
154	ATS advisory that the radar service is terminated.	RADAR SERVICE TERMINATED	N	L	R
191	ATS advisory that the aircraft is entering airspace in which no air traffic services are provided and all existing air traffic services are terminated.	ALL ATS TERMINATED	N	M	R
155	ATS advisory that radar contact has been established at the specified position.	RADAR CONTACT [position]	N	M	R
156	ATS advisory that radar contact has been lost.	RADAR CONTACT LOST	N	M	R
210	ATS advisory that the aircraft has been identified on radar at the specified position.	IDENTIFIED [position]	N	M	R
193	Notification that radar identification has been lost.	IDENTIFICATION LOST	N	M	R
157	Instruction that a continuous transmission is detected on the specified frequency. Check the microphone button.	CHECK STUCK MICROPHONE [frequency]	U	M	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
158	ATS advisory that the ATIS information identified by the specified code is the current ATIS information.	ATIS [atis code]	N	L	R
212	ATS advisory that the specified ATIS information at the specified airport is current.	[facility designation] ATIS [atis code] CURRENT	N	L	R
214	ATS advisory that indicates the RVR value for the specified runway.	RVR RUNWAY [runway] [rvr]	N	M	R
224	ATS advisory that no delay is expected.	NO DELAY EXPECTED	N	L	R
225	ATS advisory that the expected delay has not been determined.	DELAY NOT DETERMINED	N	L	R
226	ATS advisory that the aircraft may expect to be cleared to commence its approach procedure at the specified time.	EXPECTED APPROACH TIME [time]	N	L	R

Table 2.3.7-15. System Management Messages (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
159	A system generated message notifying that the ground system has detected an error.	ERROR [error information]	U	M	N
160	Notification to the avionics that the specified data authority is the next data authority. If no data authority is specified, this indicates that any previously specified next data authority is no longer valid.	NEXT DATA AUTHORITY [facility]	L	N	N
161	Notification to the avionics that the data link connection with the current data authority is being terminated.	END SERVICE	L	N	N
162	Notification that the ground system does not support this message.	SERVICE UNAVAILABLE	L	L	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
234	Notification that the ground system does not have a flight plan for that aircraft.	FLIGHT PLAN NOT HELD	L	L	N
163	Notification to the pilot of an ATSU identifier.	[facility designation]	L	N	N
227	Confirmation to the aircraft system that the ground system has received the message to which the logical acknowledgment refers and found it acceptable for display to the responsible person.	LOGICAL ACKNOWLEDGMENT	N	M	N
233	Notification to the pilot that messages sent requiring a logical acknowledgment will not be accepted by this ground system.	USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED	N	M	N

Table 2.3.7-16. Additional Messages (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
164	The associated instruction may be complied with at any future time.	WHEN READY	L	N	N
230	The associated instruction is to be complied with immediately.	IMMEDIATELY	D	H	N
165	Used to link two messages, indicating the proper order of execution of clearances/ instructions.	THEN	L	N	N
166	The associated instruction is issued due to traffic considerations.	DUE TO [traffic type] TRAFFIC	L	N	N
167	The associated instruction is issued due to airspace restrictions.	DUE TO AIRSPACE RESTRICTION	L	N	N
168	The indicated communication should be ignored.	DISREGARD	U	M	R
176	Notification that the pilot is responsible for maintaining separation from other traffic and is also responsible for maintaining visual meteorological conditions.	MAINTAIN OWN SEPARATION AND VMC	N	M	W/U

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
177	Used in conjunction with a clearance/instruction to indicate that the pilot may execute when prepared to do so.	AT PILOTS DISCRETION	L	L	N
178	(reserved)		L	L	Y
169		[free text]	N	L	R
170		[free text]	D	H	R
183		[free text]	N	M	N
187		[free text]	L	N	N
194		[free text]	N	L	Y
195		[free text]	L	L	R
196		[free text]	N	M	W/U
197		[free text]	U	M	W/U
198		[free text]	D	H	W/U
199		[free text]	N	L	N
201	Not used		L	L	N
202	Not used		L	L	N
203		[free text]	N	M	R
204		[free text]	N	M	Y
205		[free text]	N	M	A/N
206		[free text]	L	N	Y
207		[free text]	L	L	Y
208		[free text]	L	L	N

Note.— Free text message elements have no associated message intent. The capability to send a free text message with any of the attribute combinations already used in the message set has been provided.

2.3.7.9.3 Downlink message elements shall comply with the intent, use, and element attributes as presented in Tables 2.3.7-17 to 2.3.7-28.

Table 2.3.7-17. Responses (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
0	The instruction is understood and will be complied with.	WILCO	N	M	N
1	The instruction cannot be complied with.	UNABLE	N	M	N
2	Wait for a reply.	STANDBY	N	M	N
3	Message received and understood.	ROGER	N	M	N
4	Yes.	AFFIRM	N	M	N
5	No.	NEGATIVE	N	M	N

Table 2.3.7-18. Vertical Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Request to fly at the specified level.	REQUEST [level]	N	L	Y
7	Request to fly at a level within the specified vertical range.	REQUEST BLOCK [level] TO [level]	N	L	Y
8	Request to cruise climb to the specified level.	REQUEST CRUISE CLIMB TO [level]	N	L	Y
9	Request to climb to the specified level.	REQUEST CLIMB TO [level]	N	L	Y
10	Request to descend to the specified level.	REQUEST DESCENT TO [level]	N	L	Y
11	Request that at the specified position a climb to the specified level be approved.	AT [position] REQUEST CLIMB TO [level]	N	L	Y
12	Request that at the specified position a descent to the specified level be approved.	AT [position] REQUEST DESCENT TO [level]	N	L	Y
13	Request that at the specified time a climb to the specified level be approved.	AT [time] REQUEST CLIMB TO [level]	N	L	Y

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
14	Request that at the specified time a descent to the specified level be approved.	AT [time] REQUEST DESCENT TO [level]	N	L	Y
69	Request that a descent be approved on a see-and-avoid basis.	REQUEST VMC DESCENT	N	L	Y

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-19. Lateral Off-Set Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
15	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved.	REQUEST OFFSET [specified distance] [direction] OF ROUTE	N	L	Y
16	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved from the specified position.	AT [position] REQUEST OFFSET [specified distance] [direction] OF ROUTE	N	L	Y
17	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved from the specified time.	AT [time] REQUEST OFFSET [specified distance] [direction] OF ROUTE	N	L	Y

Table 2.3.7-20. Speed Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
18	Request to fly at the specified speed.	REQUEST [speed]	N	L	Y
19	Request to fly within the specified speed range.	REQUEST [speed] TO [speed]	N	L	Y

Table 2.3.7-21. Voice Contact Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
20	Request for voice contact.	REQUEST VOICE CONTACT	N	L	Y
21	Request for voice contact on the specified frequency.	REQUEST VOICE CONTACT [frequency]	N	L	Y

Table 2.3.7-22. Route Modification Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
22	Request to track from the present position direct to the specified position.	REQUEST DIRECT TO [position]	N	L	Y
23	Request for the specified procedure clearance.	REQUEST [procedure name]	N	L	Y
24	Request for a route clearance.	REQUEST CLEARANCE [route clearance]	N	L	Y
25	Request for a clearance.	REQUEST [clearance type] CLEARANCE	N	L	Y
26	Request for a weather deviation to the specified position via the specified route.	REQUEST WEATHER DEVIATION TO [position] VIA [route clearance]	N	M	Y
27	Request for a weather deviation up to the specified distance off track in the specified direction.	REQUEST WEATHER DEVIATION UP TO [specified distance] [direction] OF ROUTE	N	M	Y
70	Request a clearance to adopt the specified heading.	REQUEST HEADING [degrees]	N	L	Y
71	Request a clearance to adopt the specified ground track.	REQUEST GROUND TRACK [degrees]	N	L	Y

Table 2.3.7-23. Reports (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
28	Notification of leaving the specified level.	LEAVING [level]	N	L	N
29	Notification of climbing to the specified level.	CLIMBING TO [level]	N	L	N
30	Notification of descending to the specified level.	DESCENDING TO [level]	N	L	N
31	Notification of passing the specified position.	PASSING [position]	N	L	N
78	Notification that at the specified time, the aircraft's position was as specified.	AT [time] [distance] [to/from] [position]	N	L	N

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
32	Notification of the present level.	PRESENT LEVEL [level]	N	L	N
33	Notification of the present position.	PRESENT POSITION [position]	N	L	N
34	Notification of the present speed.	PRESENT SPEED [speed]	N	L	N
113	Notification of the requested speed.	[speed type] [speed type] [speed type] SPEED [speed]	N	L	N
35	Notification of the present heading in degrees.	PRESENT HEADING [degrees]	N	L	N
36	Notification of the present ground track in degrees.	PRESENT GROUND TRACK [degrees]	N	L	N
37	Notification that the aircraft is maintaining the specified level.	MAINTAINING [level]	N	L	N
72	Notification that the aircraft has reached the specified level.	REACHING [level]	N	L	N
76	Notification that the aircraft has reached a level within the specified vertical range.	REACHING BLOCK [level] TO [level]	N	L	N
38	Read-back of the assigned level.	ASSIGNED LEVEL [level]	N	M	N
77	Read-back of the assigned vertical range.	ASSIGNED BLOCK [level] TO [level]	N	M	N
39	Read-back of the assigned speed.	ASSIGNED SPEED [speed]	N	M	N
40	Read-back of the assigned route.	ASSIGNED ROUTE [route clearance]	N	M	N
41	The aircraft has regained the cleared route.	BACK ON ROUTE	N	M	N
42	The next waypoint is the specified position.	NEXT WAYPOINT [position]	N	L	N
43	The ETA at the next waypoint is as specified.	NEXT WAYPOINT ETA [time]	N	L	N
44	The next but one waypoint is the specified position.	ENSUING WAYPOINT [position]	N	L	N
45	Clarification of previously reported waypoint passage.	REPORTED WAYPOINT [position]	N	L	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
46	Clarification of time over previously reported waypoint.	REPORTED WAYPOINT [time]	N	L	N
47	The specified (SSR) code has been selected.	SQUAWKING [code]	N	L	N
48	Position report.	POSITION REPORT [position report]	N	M	N
79	The code of the latest ATIS received is as specified.	ATIS [atis code]	N	L	N
89	The specified ATS unit is being monitored on the specified frequency.	MONITORING [unitname] [frequency]	U	M	N
102	Used to report that an aircraft has landed.	LANDING REPORT	N	N	N
104	Notification of estimated time of arrival at the specified position.	ETA [position][time]	L	L	N
105	Notification of the alternative aerodrome for landing.	ALTERNATE AERODROME [airport]	L	L	N
106	Notification of the preferred level.	PREFERRED LEVEL [level]	L	L	N
109	Notification of the preferred time to commence descent for approach.	TOP OF DESCENT [time]	L	L	N
110	Notification of the preferred position to commence descent for approach.	TOP OF DESCENT [position]	L	L	N
111	Notification of the preferred time and position to commence descent for approach.	TOP OF DESCENT [time] [position]	L	L	N

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-24. Negotiation Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
49	Request for the earliest time at which a clearance to the specified speed can be expected.	WHEN CAN WE EXPECT [speed]	L	L	Y

	Message Intent/Use	Message Element	URG	ALRT	RESP
50	Request for the earliest time at which a clearance to a speed within the specified range can be expected.	WHEN CAN WE EXPECT [speed] TO [speed]	L	L	Y
51	Request for the earliest time at which a clearance to regain the planned route can be expected.	WHEN CAN WE EXPECT BACK ON ROUTE	L	L	Y
52	Request for the earliest time at which a clearance to descend can be expected.	WHEN CAN WE EXPECT LOWER LEVEL	L	L	Y
53	Request for the earliest time at which a clearance to climb can be expected.	WHEN CAN WE EXPECT HIGHER LEVEL	L	L	Y
54	Request for the earliest time at which a clearance to cruise climb to the specified level can be expected.	WHEN CAN WE EXPECT CRUISE CLIMB TO [level]	L	L	Y
87	Request for the earliest time at which a clearance to climb to the specified level can be expected.	WHEN CAN WE EXPECT CLIMB TO [level]	L	L	Y
88	Request for the earliest time at which a clearance to descend to the specified level can be expected.	WHEN CAN WE EXPECT DESCENT TO [level]	L	L	Y

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-25. Emergency Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
55	Urgency prefix.	PAN PAN PAN	U	H	Y
56	Distress prefix.	MAYDAY MAYDAY MAYDAY	D	H	Y
112	Indicates specifically that the aircraft is being subjected to unlawful interference.	SQUAWKING 7500	U	H	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
57	Notification of fuel remaining and number of persons on board.	[remaining fuel] OF FUEL REMAINING AND [persons on board] PERSONS ON BOARD	U	H	Y
58	Notification that the pilot wishes to cancel the emergency condition.	CANCEL EMERGENCY	U	M	Y
59	Notification that the aircraft is diverting to the specified position via the specified route due to an urgent need.	DIVERTING TO [position] VIA [route clearance]	U	H	Y
60	Notification that the aircraft is deviating the specified distance in the specified direction off the cleared route and maintaining a parallel track due to an urgent need.	OFFSETTING [specified distance] [direction] OF ROUTE	U	H	Y
61	Notification that the aircraft is descending to the specified level due to an urgent need.	DESCENDING TO [level]	U	H	Y
80	Notification that the aircraft is deviating up to the deviating distance from the cleared route in the specified direction due to an urgent need.	DEVIATING UP TO [specified distance] [direction] OF ROUTE	U	H	Y

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

Table 2.3.7-26. System Management Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
62	A system-generated message that the avionics has detected an error.	ERROR [error information]	U	L	N
63	A system-generated denial to any CPDLC Message sent from a ground facility that is not the current data authority.	NOT CURRENT DATA AUTHORITY	L	L	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
99	A system-generated message to inform a ground facility that it is now the current data authority	CURRENT DATA AUTHORITY	L	L	N
64	Notification to the ground system that the specified ATSU is the current data authority.	[facility designation]	L	L	N
107	A system-generated message sent to a ground system that tries to connect to an aircraft when a current data authority has not designated the ground system as the NDA.	NOT AUTHORIZED NEXT DATA AUTHORITY	L	L	N
73	A system-generated message indicating the software version number.	[version number]	L	L	N
100	Confirmation to the ground system that the aircraft system has received the message to which the logical acknowledgment refers and found it acceptable for display to the responsible person.	LOGICAL ACKNOWLEDGMENT	N	M	N

Table 2.3.7-27. Additional Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
65	Used to explain reasons for pilot's message.	DUE TO WEATHER	L	L	N
66	Used to explain reasons for pilot's message.	DUE TO AIRCRAFT PERFORMANCE	L	L	N
74	States a desire by the pilot to provide his/her own separation and remain in VMC.	REQUEST TO MAINTAIN OWN SEPARATION AND VMC	L	L	Y
75	Used in conjunction with another message to indicate that the pilot wishes to execute request when the pilot is prepared to do so.	AT PILOTS DISCRETION	L	L	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
101	Allows the pilot to indicate a desire for termination of CPDLC-service with the current data authority.	REQUEST END OF SERVICE	L	L	Y
103	Allows the pilot to indicate that he/she has cancelled IFR flight plan.	CANCELLING IFR	N	L	Y
108	Notification that de-icing action has been completed.	DE-ICING COMPLETE	L	L	N
67		[free text]	N	L	N
68		[free text]	D	H	Y
90		[free text]	N	M	N
91		[free text]	N	L	Y
92		[free text]	L	L	Y
93		[free text]	U	H	N
94		[free text]	D	H	N
95		[free text]	U	M	N
96		[free text]	U	L	N
97		[free text]	L	L	N
98		[free text]	N	N	N

Note.— Free text message elements have no associated message intent. The capability to send a free text message with any of the attribute combinations already used in the message set has been provided.

Table 2.3.7-28. Negotiation Responses (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
81	We can accept the specified level at the specified time.	WE CAN ACCEPT [level] AT [time]	L	L	N
82	We cannot accept the specified level.	WE CANNOT ACCEPT [level]	L	L	N
83	We can accept the specified speed at the specified time.	WE CAN ACCEPT [speed] AT [time]	L	L	N

	Message Intent/Use	Message Element	URG	ALRT	RESP
84	We cannot accept the specified speed.	WE CANNOT ACCEPT [speed]	L	L	N
85	We can accept a parallel track offset the specified distance in the specified direction at the specified time.	WE CAN ACCEPT [specified distance] [direction] at [time]	L	L	N
86	We cannot accept a parallel track offset the specified distance in the specified direction.	WE CANNOT ACCEPT [specified distance] [direction]	L	L	N

Note.— Wherever the variable "level" is specified, the message can specify either a single level or a vertical range, i.e. block level.

2.3.7.10 Parameter Value Unit, Range, and Resolution

2.3.7.10.1 A CPDLC-user shall interpret CPDLC Message element variables unit, range, and resolution as defined in 2.3.4.

2.3.8 SUBSETTING RULES

2.3.8.1 General

Note.— This chapter specifies conformance requirements which all implementations of the CPDLC protocol obey.

2.3.8.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service claiming conformance to 2.3 shall support the CPDLC protocol features as shown in the tables below:

Note.— The 'status' column indicates the level of support required for conformance to the CPDLC-ASE protocol described in this part. The values are as follows:

'M' *mandatory support is required,*

'O' *optional support is permitted for conformance to the CPDLC protocol,*

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

'C.n' *the item is conditional where n is the number which identifies the condition which is applicable. The definitions for the conditional statements used in this chapter are written under the tables in which they first appear.*

Table 2.3. 8-1. Protocol Versions Implemented

	Status	**Associated Predicate**
Version 1	M	none

Table 2.3.8-2. CPDLC Protocol Options

	Status	**Associated Predicate**
CPDLC-air-ASE	C.1	CPDLC/air
CPDLC-ground-ASE	C.1	CPDLC/ground
DSC function supported	if (CPDLC/air) O, else M	DSC-FU
DSC function supported by CPDLC-ground-user	if (CPDLC/ground) O, else N/A	DSC-USER
Forward function supported by initiating user	if (CPDLC/ground) O, else N/A	FWD-INIT
Forward function supported by receiving user	if (CPDLC/ground) O, else N/A	FWD-USER

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

Table 2.3.8-3. CPDLC-air-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I.	CPDLC/air	a CPDLC-air-ASE supporting just the core* CPDLC functionality (no downstream clearance capability)
II.	CPDLC/air + DSC-FU	a CPDLC-air-ASE supporting the core CPDLC functionality and the downstream clearance capability (complete CPDLC-air-ASE functionality)
		* the core CPDLC functionality is defined as support for the CPDLC-start, message, end services plus abort services.

Table 2.3.8-4. CPDLC-ground-ASE Conformant Configurations

	List of Predicates	**Functionality Description**
I.	CPDLC/ground	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for receiving and indicating that the forward function is not supported
II.	CPDLC/ground + DCS-FU + DSC-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for receiving and indicating that the ground forward function is not supported
III.	CPDLC/ground + DSC-FU + FWD-INIT	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for receiving and indicating that the ground forward function is not supported • functionality for supporting the capability to initiate the ground forwarding of CPDLC Messages
IV.	CPDLC/ground + DCS-FU + DSC-USER + FWD-INIT	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for receiving and indicating that the ground forward function is not supported • functionality for supporting the capability to initiate the ground forwarding of CPDLC Messages

	List of Predicates	**Functionality Description**
V.	CPDLC/ground + DSC-FU FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for DSC dialogue.
VI.	CPDLC/ground + DCS-FU + DSC-USER + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for CPDLC-ground-user support for the receipt of CPDLC ground forward messages
VII.	CPDLC/ground + DSC-FU + FWD-INIT + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • full CPDLC ground forwarding functionality (initiating and user receiving)
VIII.	CPDLC/ground + DSC-FU + DSC-USER + FWD-INIT + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • full CPDLC ground forwarding functionality (initiating and user receiving) (complete CPDLC-ground-ASE functionality)