ATNP/WG3/SG2 (Air/Ground Application)
Proposed Version 3.0 (Brussels)
22 April 1996

AERONAUTICAL TELECOMMUNICATION NETWORK PANEL

WORKING GROUP 3 (APPLICATIONS AND UPPER LAYERS)

Sub Group 2 (Air/Ground Applications)

Part 3: Controller Pilot Data Link Communication Application SARPs

Prepared by: Air/Ground Applications Subgroup

SUMMARY

This document is the draft of the Controller/Pilot Data Link Communication Application SARPs for the ATN CNS/ATM-1 Package.

CONFIGURATION SHEET

Title : Draft CPDLC SARPs for the ATN CNS/ATM-1 Package

Version : 3.0 Proposed

Date: 22 April 1996

Contact: Airservices Australia, Eurocontrol, Federal Aviation Administration, Direction Generale de l'Aviation Civil, UK National Air Traffic Services

Status: Draft

Change History :

Version	Description	Affected Parts	Date
1.0	Banff Proposed Draft CPDLC SARPs		6 Oct. 1995
1.1	Input to ATNP/WG3 at Brisbane		5 Feb. 1996
	Chapter 1		
	• addition of DSC functionality	§1.1.4, §1.2.2, §1.3.3	
	• clarifications of AE/ASE definitions	§1.1.4	
	Chapter 2		
	• Time accuracy clarified	§2.2.1	
	Chapter 3		
	• DSC changes		
	• DSC-start service	§3.3.1, new §3.8	
	• DSC-end service	§3.3.1, new §3.9	
	Chapter 4		
	• DSC addition	§4.1	
	• high level ASN.1 change to permit DSC mode (DownlinkPDUs definition), Mode, and StartdownMessage		
	Chapter 5		
	• DSC addition		
	• additional time sequence diagrams	§5.1.4, §5.17	
	• addition of timer for DSC	§5.2.2.1	
	• addition of DSC mode	§5.3.1.4	
	• modifications to D-START confirmation	§5.3.3.1, §5.3.3.2	
	• modifications to D-START indication	§5.3.2.1, §5.3.2.3	
	• modifications to D-END indication	§5.3.5.2	
	• modifications to D-END confirmation	§5.3.6.2	
	• modifications to CPDLC-start (mode)	§5.3.7.1.a	
	• addition of DSC-start request	§5.3.9	
	• addition of DSC-start response	§5.3.10	
	• addition of DSC-end request	§5.3.13	
	• addition of DSC-end response	§5.3.15	
	• Change from D-U-ABORT to D-ABORT plus parameters	§	
	• D-ABORT modified sequence diagrams	§5.1.9, §5.1.10	
	• modified user-abort	§5.3.17	
	• modified D-ABORT	§5.3.18	
	• Due to modified D-ABORT (new name, new variables) all of section 5.4 was deleted and rewritten.	§5.4	

	Chapter 6		
	• Addition of note for ATSC value references	§6.2.2.3.1	
	Chapter 7		
	• Freeing of message identification		
	• DSC modifications		
	• CPDLC/DSC distinction	§7.3.3	
	• CPDLC-start service modifications	§7.4.2.1	
	• addition of DSC-start service section	§7.5.2	
	• modification of CPDLC-message indication		
	• addition of DSC-end service	§7.4.5	
	• Logical Acknowledgment both ways	§7.3.5,	
	• Initiation/closure manual/data link	§7.3.7 notes	
	• Clarification of Response requirements	§7.3.7	
	• receipt of start when a connection is assumed (one side has crashed and other side is not yet aware)	§7.4.3.1, §7.4.3.2, §7.5.3.1	
	Appendix A		
	• State tables updated for DSC	§A	
	• State tables updated for D-ABORT Change		
2.0	Output from Brisbane - Draft SARPs		15 Feb. 1996
	Chapter 1		
	• addition of ground forward capability	§1.2.2.1, §1.3.4	
	• new section for ground/ground ATN peers	§1.1.2.2, §1.1.4.2	
	• modified header for ground/ground	§1.4.2, §1.4.3	
	• Changed reference of traffic type values to “[5]”	§1.1.5.1	
	• modification to message response tables from ADSP San Francisco	§1.4.6.5	
	• removal of appendix references	§1.3.3, §1.4.6.6	
	Chapter 2		
	• Timing requirements clarified	§2.2	
	Chapter 3		
	• Change of ICAO Facility Designator from 4 to 8 characters	§3.5.2.2, §3.5.3.1, §3.6.2.1	
	• Insertion of reference [5] for Class of Communications	§3.5.7.1, §3.6.7.1	
	• Addition of CPDLC-forward service	§3.3.1, new §3.10	
	Chapter 4		
	• Automatic tagging/removing of tags in ASN.1		
	• Alphabetizing ASN.1		
	• Addition of forward service		
	• Renaming DownlinkPDUs to AircraftPDUs, UplinkPUs to GroundPDUs	§4.1	
	• Addition of ATCForwardMessage, ForwardHeader, and ForwardMessage ASN.1 types	§4.1	
	• ICAO facility designator modified from 4 character to 8	§4.1	
	• Change of Pre-departure clearance to departure clearance	§4.1	
	• Additional message variables (clearance type, speed type)	§4.1	
	Chapter 5		
	• Addition of CPDLC-forward service		
	• New sequence diagrams for forward	§5.1.8, §5.1.11, §5.1.12	
	• New timers for forward	§5.2.2.1	
	• Modifications to Protocol Descriptions		
	• New CPDLC forward request	§5.3.16	
	• Modifications to D-START Indication	§5.3.2.2	

• Modifications to D-START Confirmation	§5.3.3.3
• Modifications to state tables	§5.5
• General re-ordering of section 5.3, so the dialogue service primitives are defined first followed by the CPDLC service primitives (for both air and ground protocol descriptions)	§5.3
• State tables moved from Appendix A to section 5.5	§5.5
Chapter 6	
• NULL removal from QOS table	§6.2.2.3.1, Table 6-1
• Addition of RER clarification (Note 2)	§6.2.2.3.1
Chapter 7	
• Modifications for forward function	§
• differentiation of ground/ground from air/ground messages	§7.3.4
• differentiation of air/ground ground/ground message headers	§7.2.1, §7.2.2
• differing air/ground ground/ground response requirements	§7.3.7.19
• addition of sections for forward service	§7.3.7.19
• CPDLC-start service modification	§7.5.1.1.1
• new forward service section	§7.5.5
• Duplicate message id causes abort not error	§7.3.8.1.1
• Clarification of YES response requirements	§7.3.7.13, §7.3.7.15
• Only One NDA	§7.4.3.1.2, §7.4.3.1.3, §7.4.3.1.4
• Moving of Appendix B to Chapter 7	§7.6
• Insertion of Range and Resolution Message Variables	§7.7
Appendix A	
• State table moved to Chapter 5; appendix A deleted	§A
Appendix B	
• Message Descriptions Moved to section 7.6, appendix B deleted	§B

3.0	**Input to ATNP/WG3 at Brussels**		**5 Feb. 1996**
	Chapter 1		
	• Airframe and aircraft ID needed in ground/ground header (see CPDLC defect 10)	§1.4.3	
	• Generalizing RCP (see CPDLC defect 13)	§1.1.4.1.1	
	• Updating of references (see CPDLC defect 14)	§1.1.5	
	Chapter 2		
	• Generalizing RCP (see CPDLC defect 13)	§2.1	
	Chapter 3		
	• Refinement in use of shalls/notes	§3.3.1, 3.3.2	
	• Tightening of use of CPDLC message in CPDLC-start or DSC-start response	§3.5, 3.6	
	• Addition of “I” and “J” routing classes (CPDLC defect 17)	§3.5, 3.6, 3.10	
	• Clarification that data could be lost in a user abort (CPDLC 18)	§3.11	
	Chapter 4 (ASN.1)		
	• Re-insertion of Tags (see CPDLC defect 1)	§4	
	• Fix of miscellaneous ASN.1 errors (see CPDLC defect # 2, 3, 4, 12)	§4	
	• Addition of Unlawful interference messages (see CPDLC defect 6)	§4	
	• Additional CRUISE message (see CPDLC defect 7)	§4	
	• Modifications to message range and resolutions (see CPDLC defect 8)	§4	
	• Change from souls to persons on board (see CPDLC defect 9)	§4	
	• Airframe and aircraft ID needed in ground/ground header (see CPDLC defect 10)	§4	
	• Change in alerting requirements for message 223 (see CPDLC defect 11)	§4	
	Chapter 5		
	• Clarification of timer requirements (CPDLC defect 19)	§5.3	
	Chapter 6		
	• Addition of “I” and “J” routing classes (CPDLC defect 17)	§6.2.2.3	
	Chapter 7		
	• Correcting message 197/192 labeling error (see CPDLC defect 5)	§7.7	
	• Addition of Unlawful interference messages (see CPDLC defect 6)	§7.7	
	• Additional CRUISE message (see CPDLC defect 7)	§7.7	
	• Change from souls to persons on board (see CPDLC defect 9)	§7.7	
	• Change in alerting requirements for message 223 (see CPDLC defect 11)	§7.7	
	• Addition general requirements added to chapter 7 (see CPDLC defect 20)	§7.1	

	• User response timing requirements (see CPDLC defect 21)	§7.5.1.2, 7.5.2.2, 7.5.4.2, 7.5.5.2	
	• DSC-start and DSC-end indication requirements (see CPDLC defect 22)	§7.5.2.2, 7.5.5.5	

TABLE OF CONTENTS

1. APPLICATION OVERVIEW

1.1 Introduction

1.1.1 Purpose

1.1.1.1 The purpose of this document is to define draft Standards and Recommended Practices (SARPs) for the Controller Pilot Data Link Communication (CPDLC) application through the use of the Aeronautical Telecommunication Network (ATN). This application allows data link communication between controllers and pilots. The ATN provides the media and protocols to conduct data link communication for the CPDLC application.

1.1.1.2 Compliance with these standards is a means of assuring that the Air Traffic Control (ATC) system will perform its intended functions using data link, and that the CPDLC application is implemented in a globally uniform and interoperable manner.

1.1.2 Background

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

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

1.1.2.3 In the performance of this role, the CPDLC application interacts with the following:

a) Aeronautical Telecommunication Network (ATN),
b) Context Management (CM) Application,
c) Ground ATC systems, and
d) The aircraft avionics.

1.1.2.4 The CPDLC application provides air-ground and ground-ground data communications for ATC service. The CNS/ATM-1 service defined in this document includes a set of clearance/information/request message elements which correspond to existing phraseology employed by current Air Traffic Control procedures. The controller is provided with the capability to issue altitude assignments, crossing constraints, lateral deviations, route changes and clearances, speed assignments, radio frequency assignments, and various requests for information, as well as forward CPDLC messages to another controller. The pilot is provided with the capability to respond to messages, to request clearances and information, to report information, and to declare/rescind an emergency. A "free text" capability is also provided to exchange information not conforming to defined formats.

1.1.2.5 Controllers and pilots will use data link services to augment the existing voice communication. It is expected to be used for routine or frequent types of transactions. Although initial implementation is intended to conform to existing procedures, it is anticipated that future evolution of the system and procedures will result in the greater automation of functions for both aircraft and ground systems.

1.1.3 Structure of Document

1.1.3.1 Chapter 1: APPLICATION OVERVIEW contains the document’s purpose and structure, a summary of the ADSP operational requirements (ORs) that relate to CPDLC and maps these ORs to the functions of CPDLC.

1.1.3.2 Chapter 2: GENERAL REQUIREMENTS contains performance, time accuracy, security, backwards compatibility, and error processing requirements.

1.1.3.3 Chapter 3: ABSTRACT SERVICE DEFINITION contains the description of the abstract service provided by the CPDLC Application Service Element (CPDLC-ASE).

1.1.3.4 Chapter 4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by CPDLC ASEs using Abstract Syntax Notation Number One (ASN.1).

1.1.3.5 Chapter 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.

1.1.3.6 Chapter 6: COMMUNICATION REQUIREMENTS contains the requirements that the CPDLC application imposes on the underlying communication system.

1.1.3.7 Chapter 7 : CPDLC USER REQUIREMENTS contains requirements imposed on the user of the CPDLC ASE service and message description tables.

1.1.4 Explanation of Terms

1.1.4.1 Acronyms

1.1.4.1.1 The following abbreviations are used in this document:

ADS	Automatic Dependent Surveillance
ADSP	Automatic Dependent Surveillance Panel
AE	Application Entity
APDU	Application Protocol Data Unit
ASE	Application Service Element
ASN.1	Abstract Syntax Notation Number One
ATC	Air Traffic Control
ATM	Air Traffic Management
ATN	Aeronautical Telecommunication Network
ATS	Air Traffic Service
ATSC	Air Traffic Services Communication
CF	Control Function
CM	Context Management
CNS	Communication Navigation Surveillance
CNS/ATM	Communications Navigation Surveillance / Air Traffic Management
CPC	Controller Pilot Communications
CPDLC	Controller Pilot Data Link Communications
DS	Dialogue Service
DSC	Down Stream Clearance
FD	Functional Description
FDPS	Flight Data Processing System
IA5	International Alphabet Number 5
ICAO	International Civil Aviation Organization
ID	Identification
IEC	International Electrotechnical Committee
IS	International Standard
ISO	International Organization for Standardization
ITU	International Telecommunications Union
OR	Operational Requirement
PDU	Protocol Data Unit
PER	Packed Encoding Rules
QOS	Quality of Service
~~**RCP**~~	~~Required Communication Performance~~

RER	Residual Error Rate
SARPs	Standards and Recommended Practices
UTC	Coordinated Universal Time

1.1.4.2 General Definitions

1.1.4.2.1 For the purpose of this document, the following definitions apply:

a) ***Active user:*** the user is currently involved in a CPDLC dialogue.

b) ***AE Qualifier:*** that part of the AE title that uniquely identifies the particular application entity.

c) ***AE Title:*** a unique name for an application entity.

d) ***APDU:*** basic unit of information exchanged between two CPDLC ASEs.

e) ***Application Entity:*** a model of those aspects of an application process that are significant from the viewpoint of accessing OSI capabilities.

f) ***Application Process:*** an element within an open system which performs information processing tasks for a particular application.

g) ***ASE*:** an abstract module of a system providing service to other parts of the system.

h) ***Context Management*:** an independent service that meets ATSC addressing requirements. It provides the mechanism for aircraft and ATC ground systems to indicate availability to other ATN users and to convey the addresses to be employed. The aircraft CM application interfaces with aircraft equipment to provide ATC ground system the addresses needed to establish communication with the aircraft.

i) ***CPDLC AE abstract service interface:*** the abstract interface between the CPDLC-users and the CPDLC-service provider.

j) ***CPDLC ASE abstract service interface:*** the abstract interface through which the CM-ASE service are accessed.

Note: — In version 1 of the CPDLC application, this interface coincide with the CPDLC-AE abstract service interface.

k) ***CPDLC-air-ASE:*** an abstract part of the aircraft system which performs the communication related functions of CPDLC.

l) ***CPDLC-air-user:*** the abstract part of the aircraft system which performs the non-communication related functions of CPDLC.

m) ***CPDLC-CF:*** that abstract part of the application entity which performs the mapping between the CPDLC-ASE service primitives and other elements within the CPDLC application.

n) ***CPDLC-ground-ASE:*** an abstract part of the ground system which performs the communication related functions of CPDLC.

o) ***CPDLC-ground-user:*** the abstract part of the ground system which performs the non-communication related functions of CPDLC.

p) ***CPDLC service primitive:*** a function of a CPDLC-AE that is not broken down further into sub-functions, and is presented as part of the CPDLC-AE abstract service interface (i.e., request, indication, response, or confirmation).

q) ***CPDLC service provider:*** the CPDLC-service provider is composed of the ground and airborne CPDLC AEs, all underlying data communication protocol entities and the physical media. As a consequence, it encompasses everything between the CPDLC-AE service interfaces of the end-users of the CPDLC application.

r) ***Current Data Authority:*** the ground system which is technically permitted to conduct a CPDLC dialogue with an aircraft.

s) ***Dialogue Service:*** the service which allows the CPDLC-air-ASE to communicate with the CPDLC-ground-ASE and vice-versa, or allows a CPDLC-ground-ASE to communicate with another CPDLC-ground-ASE.

t) ***Downstream Data*** *Authority*: the ground system which is technically permitted to conduct a DSC dialogue with an aircraft.

u) ***Message***: information exchanged between the CPDLC air-user and the CPDLC-ground-user.

v) ***Message Element:*** a component of a message used to define the context of the information exchanged.

w) ***Message Element Identifier:*** the ASN.1 tag of the ATCUplinkMsgElementID or the ATCDownlinkMsgElementId.
x) ***Message Header (air/ground):*** control information used to maintain synchronization between the aircraft and the ground ATC system.
y) ***Message Header (ground/ground):*** control information used to maintain synchronization between the two ground ATC systems.
z) ***Message Identification Number:*** a unique number assigned to each air/ground message. This number is used to differentiate messages and is conveyed in an air/ground message header.
aa) ***Message Reference Number:*** used to uniquely associate a response with a previously received message. The Message Identification Number of a previously received message becomes the Message reference number of the response message. The Message Reference number is conveyed in the message header.
bb) ***Next Data Authority:*** the ground system so designated by the Current Data Authority.
cc) ***Residual Error Rate:*** the ratio of messages mis-delivered, non-delivered, or delivered with an error undetected by the system, over the total number of messages delivered to the system.

1.1.4.3 Conventions For Expressing Requirements

1.1.4.3.1 The following conventions apply for expressing requirements in this document:

a) ***shall*** **-** used to state a mandatory requirement.
b) ***should*** **-** used to state a recommended practice.

1.1.5 References

1.1.5.1 The following references are used in this document:

[1] Draft ICAO Manual of Air Traffic Services (ATS) Data Link Applications, 1 March 1996.~~Automatic Dependent Surveillance (ADS) and Air Traffic Services (ATS) Data Link Applications Guidance Material, November 1994.~~
[2] ISO/IEC IS 8825-2, ITU-T Recommendation X.691, Information Technology - ASN.1 Encoding Rules - Packed Encoding Rules (PER).
[3] ISO/IEC 8824-1, ITU-T Recommendation X.682, Information Technology - Abstract Syntax Notation One (ASN.1).
[4] Draft SARPs for ATN Upper Layers for CNS/ATM-1 Package, Version 2.0, 9 February 1996.
[5] Reference Document for Traffic Types~~QOS Levels reference.~~

1.2 Application Functionality

1.2.1 This section lists the operational requirements (OR) as defined by the ADSP that CPDLC addresses:~~.~~

a) Comparison of four-dimensional (4-D) profile stored in the aircraft system with flight data stored in the FDPS.
b) Approval for a flight to enter ADS-ATC airspace
c) Confirmation that the aircraft’s projected profile coincides with that stored in the FDPS.
d) Automatic transfer of control and communications between ADS-ATC airspaces using digital data interchange.
e) Airspace automatic transfer of control and communications from ADS-ATC to non-ADS-ATC airspace using digital data interchange.
f) Provision of Controller-Pilot Data Link Communications (CPDLC).

1.2.2 Mapping of Operational Requirements to Functional Descriptions

1.2.2.1 This section presents how each CPDLC related OR listed above can be mapped to the functions described in Section 1.3. Table 1-1 shows which functional descriptions assist in achieving each OR and are labeled as follows (section reference in parentheses):

a) FD1: Controller-Pilot Message Exchange (Section 1.3.1),

b) FD2: Transfer of Data Authority (Section 1.3.2),
c) FD3: Down Stream Clearance Function (Section 1.3.3), and
d) FD4: Ground Forward Function (Section 1.3.4).

Operational Requirement	FD1	FD2	FD3	FD4
Comparison of four-dimensional profile stored in the aircraft system with flight data stored in the FDPS	√			
Approval for a flight to enter ADS-ATC airspace	√	√	√	√
Confirmation that the aircraft's projected profile coincides with that stored in the FDPS	√			
Automatic transfer of control and communications between ADS-ATC airspaces using digital data interchange	√	√		√
Airspace automatic transfer of control and communications from ADS-ATC to non-ADS-ATC airspace using digital data interchange	√	√		
Provision of controller-pilot data link communications	√			

Table 1-1: OR to Functional Description Mapping

1.3 Functional Descriptions

1.3.1 FD1: Controller-Pilot Message Exchange

1.3.1.1 Functional Description

1.3.1.1.1 The controller-pilot message exchange function defines a method for a controller and pilot to exchange messages via data link. This function provides messages for the following :

a) general information exchange;
b) clearance
    1) delivery,
    2) request, and
    3) response;
c) altitude/identity surveillance;
d) monitoring of current/planned position;
e) advisories
    1) request and
    2) delivery;
f) system management functions; and
g) emergency situations.

1.3.2 FD2: Transfer of Data Authority

1.3.2.1 Functional Description

1.3.2.1.1 The Transfer of Data Authority Functions 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.

1.3.3 FD3: Down Stream Clearance (DSC) Function

1.3.3.1 Functional Description

1.3.3.1.1 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).

1.3.4 FD4: Ground Forward Function

1.3.4.1 Functional Description

1.3.4.1.1 The Ground Forward Function provides the capability for a ground system to forward information received in a CPDLC message to another ground system. This information is exchanged using CPDLC message(s).

1.4 A CPDLC Message

1.4.1 A CPDLC message is composed of a message header, and from one to five message elements.

1.4.2 For air/ground messages, the message header is composed of a message identification number, a message reference number, if required, a time stamp, and a logical acknowledgment requirement (optional).

1.4.3 For ground/ground messages, the message header is composed of a time stamp, ~~and~~ the aircraft flight identification, and the airframe identification to which the message refers.

1.4.4 A message element consists of a message element identifier, data as indicated by the specified message element, and associated message element attributes.

1.4.5 Message Identification Numbers

1.4.5.1 A message identification number pertains to a single peer to peer dialogue.

1.4.5.2 Message identification numbers used by a CPDLC ground system for uplink messages to an aircraft have no relationship to the message identification numbers used by the same ground system another aircraft.

1.4.5.3 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.

1.4.5.4 There is no relationship between message identification numbers assigned and managed by a the CPDLC ground system and those message identification numbers assigned and managed by the aircraft.

1.4.6 Message Attributes

1.4.6.1 Message attributes dictate certain message handling requirements for the CPDLC-user receiving a message. The CPDLC messages have Urgency, Alert, and Response attributes.

1.4.6.2 Attribute Association

1.4.6.2.1 Each message element has associated Urgency, Alert, and Response attributes as specified below.

1.4.6.2.2 When a message contains a single message element, the message attributes are the message element attributes.

1.4.6.2.3 When a message contains multiple message elements, the highest precedence message element attribute type associated with any element in the message become the message attribute type for the entire message. Attribute type precedence is indicated below. Message element attribute table entries are listed in order of precedence (i.e., a precedence value of 1 is highest followed by 2, etc.). For example, this means that a message containing multiple message elements, where at least one element has a W/U attribute, has a W/U attribute for the whole message.

1.4.6.3 Urgency

1.4.6.3.1 The Urgency (URG) attribute delineates the queuing requirements for received messages that are displayed to the end-user. Urgency types are presented in Table 1-2. These Urgency attribute types are used for both air/ground and ground/ground messages.

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

Table 1-2: Urgency Attribute (Air/Ground and Ground/Ground)

1.4.6.4 Alert

1.4.6.4.1 The alert (ALRT) attribute delineates the type of end-user alerting required by the CPDLC-user upon message receipt. Alert types are presented in Table 1-3. These Urgency attribute types are used for both air/ground and ground/ground messages.

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

Table 1-3: Alert Attribute (Air/Ground and Ground/Ground)

1.4.6.5 Response

1.4.6.5.1 The response (RESP) attribute mandates CPDLC-user response requirements for a given message element. Response types are presented in Table 1-4 for uplink messages and Table 1-5 for downlink messages. Response message attribute do not apply to ground/ground messages.

Type	Response Required	Valid Responses Description	Precedence
W/U	Yes	Response required: WILCO, UNABLE, STANDBY permitted, LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary)	1
A/N	Yes	Response required: AFFIRM, NEGATIVE, STANDBY permitted, LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary)	2
R	Yes	Response required: ROGER, UNABLE, STANDBY permitted LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary)	3
Y	Yes	Any CPDLC downlink message, LOGICAL ACKNOWLEDGMENT (only if required),	4
N	No, unless logical acknowledgment required	LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary, only when logical acknowledgment is required)	5

Table 1-4: Response Attribute (Up-Link)

Type	Response Required	Valid Responses Description	Precedence
Y	Yes	Any CPDLC uplink message LOGICAL ACKNOWLEDGMENT (only if required),	1
N	No, unless logical acknowledgment required	LOGICAL ACKNOWLEDGMENT (only if required), ERROR (if necessary, only when logical acknowledgment is required)	2

Table 1-5: Response Attribute (Down-Link)

1.4.6.6 See Chapter 7 for detailed CPDLC message intent/use descriptions.

2. GENERAL REQUIREMENTS

2.1 Performance Requirements

Note: — Systems developed to support CPDLC functionality will be capable of meeting the ~~Required~~ c~~C~~ommunication requirements~~ Performance (RCP)~~ appropriate for the phase of operation as specified in [5].

2.2 Time Accuracy Requirements

2.2.1 Absolute times which are sent as parameters over the data link shall be as accurate as the required resolution of the time parameter.

2.2.2 Absolute times shall be UTC.

2.2.3 Dates shall be expressed as UTC date (i.e., the day increments at 2359:59z).

2.2.4 Relative times shall be accurate to ± 0.1 second.

2.2.5 Where time stamps are used they shall consist of year, month, day and hour, minute, second.

2.3 Security Requirements

Note: — There are no internationally approved operational requirements relating to data link application security.

2.4 Backwards Compatibility Requirements

Note: — This document describes the version 1 of the CNS/ATM-1 Package CPDLC application. Best efforts will be made to ensure that subsequent versions of this protocol are backwards compatible.

2.4.1 For CNS/ATM-1 the CPDLC-air-ASE and CPDLC-ground-ASE version numbers shall both be set to one.

2.5 Error Processing Requirements

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

2.5.2 In the event of a user invoking a CPDLC service primitive when the CPDLC-ASE is not in a state specified in chapter 5, the following shall occur:

a) the invocation is rejected, and
b) the user is notified.

3. THE ABSTRACT SERVICE

3.1 Introduction

Note 1: — 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 2: — This chapter defines the static behaviour (i.e., the format) of the CPDLC abstract service. Its dynamic behaviour (i.e., how it is used) is described in Chapter 7.

3.2 The CPDLC Functional Model

Figure 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 interface,
c) the CPDLC-AE,
d) the CPDLC Control Function (CPDLC-CF),
e) the CPDLC Application Service Element (CPDLC-ASE) service interface,
f) the CPDLC-ASE, and
g) the Dialogue Service (DS) interface.

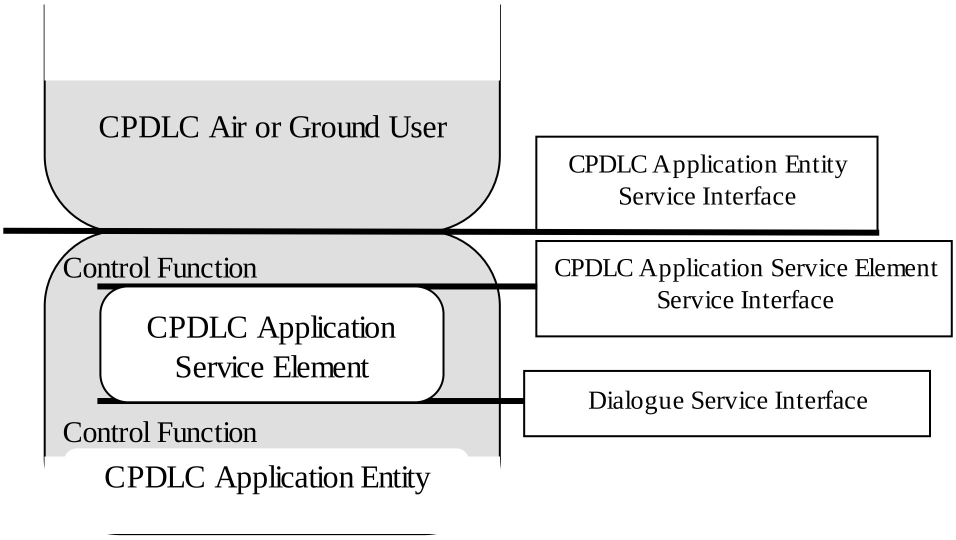
CPDLC Air or Ground User
CPDLC Application Entity Service Interface
Control Function
CPDLC Application Service Element Service Interface
CPDLC Application Service Element
Dialogue Service Interface
Control Function
CPDLC Application Entity

Figure 3-1: Functional Model of the CPDLC Application

Note 2: — 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 interface. The individual actions at this interface are called CPDLC-AE service primitives. Similarly, individual actions at other interfaces in the communication system are called service primitives at these interfaces.

Note 3: — The CPDLC-AE consists of several elements including the CPDLC-ASE and the CPDLC-CF. The DS interface is made available by the CPDLC-CF to the CPDLC-ASE for communication with the peer CPDLC-ASE.

Note 4: — 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 actions, message creation, timer management, error and exception handling.

Note 5: — The CPDLC-ASE interfaces only with the CPDLC-CF. This CPDLC-CF is responsible for mapping service primitives received from one element (such as the CPDLC-ASE and the CPDLC-user) to other elements which interface with it. The part of the CPDLC-CF which is relevant from the point of view of these SARPs, i.e. the part between the CPDLC-user and the CPDLC-ASE, will map CPDLC-AE service primitives to CPDLC-ASE service primitives transparently in the CNS/ATM-1 Package.

Note 6: — The DS interface is the interface between the CPDLC-ASE and the part of CPDLC-CF underneath the CPDLC-ASE, and provides a generic dialogue service [4].

3.3 The CPDLC-ASE Abstract Service

3.3.1 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: — 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 an interface has been built.

3.3.2 <u>The CPDLC-ASE abstract service shall consist of the following:</u>

a) *CPDLC-start service* as defined in section 3.5,
b) *DSC-start service* as defined in section 3.6,
c) *CPDLC-message service* as defined in section 3.7,
d) *CPDLC-end service* as defined in section 3.8,
e) *DSC-end service* as defined in section 3.9,
f) *CPDLC-forward service* as defined in section 3.10,
g) *CPDLC-user-abort service* as defined in section 3.11, and
h) *CPDLC-provider-abort service* as defined in section 3.12.

3.4 Conventions

Note: 1 — For a given primitive, the presence of each parameter is described by one of the following values in the parameter tables in chapter 3.

a)	***blan**k*	*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.

3.5 CPDLC-start Service

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

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

3.5.1 The CPDLC-start service primitives shall contain the parameters as presented Table 3-1.

Parameter Name	Req	Ind	Rsp	Cnf
Called Peer Identifier	M			
Calling Peer Identifier	M	M(=)		
CPDLC Message	U	C(=)		
Reject Reason			C	C(=)
Result			M	M(=)
Class of Communication Service	U			

Table 3-1: CPDLC-start Service Parameters

3.5.2 Called Peer Identifier

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

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

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

3.5.2.2 If the service is ground initiated, the *Called Peer Identifier* parameter value shall conform to the abstract syntax eight-character ICAO facility designator.

3.5.3 Calling Peer Identifier

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

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

3.5.3.1 If the service is ground initiated, the *Calling Peer Identifier* parameter value shall conform to the abstract syntax eight-character ICAO facility designator.

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

3.5.4 CPDLC Message

Note: — The CPDLC-user can use this parameter to send a CPDLC message to its peer user.

3.5.4.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage, if supplied by the CPDLC-ground-user.

3.5.4.2 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage, if supplied by the CPDLC-air-user.

3.5.5 Reject Reason

Note: — This parameter is used to provide a reason for rejecting a CPDLC dialogue.

3.5.5.1 If, and only if, the CPDLC-user rejects the request to open a CPDLC dialogue, the CPDLC user shall provide a reason (a CPDLC message) for the rejection.

3.5.5.2 The *Reject Reason* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage if supplied by the CPDLC-ground-user.

3.5.5.3 The *Reject Reason* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage if supplied by the CPDLC-air-user.

3.5.6 Result

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

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

3.5.7 Class of Communication Service

Note: — 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.

3.5.7.1 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”, “I”, or “J”.

Note: — Class of Communication service parameter values are detailed in [5].

3.6 DSC-start Service

Note 1: — The DSC-start service is used to establish a DSC dialogue for the purpose of providing down stream clearances. It is a confirmed service.

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

3.6.1 The DSC-start service primitives shall contain the parameters as presented Table 3-2.

Parameter Name	Req	Ind	Rsp	Cnf
ICAO Facility Designator	M			
Aircraft Identifier	M	M(=)		
CPDLC Message	U	C(=)		
Reject Reason			C	C(=)
Result			M	M(=)
Class of Communication Service	U			

Table 3-2: DSC-start Service Parameters

3.6.2 ICAO Facility Designator

Note: — This parameter contains the addressed ground system’s ICAO facility designator.

3.6.2.1 The *ICAO Facility Designator* parameter value shall conform to the abstract syntax eight-character ICAO facility designator.

3.6.3 Aircraft Identifier

3.6.3.1 The *Aircraft Identifier* parameter value shall conform to the abstract syntax 24-bit aircraft-id.

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

3.6.4 CPDLC Message

Note: — The CPDLC-air-user can use this parameter to send a CPDLC message to a CPDLC-ground-user.

3.6.4.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage.

3.6.5 Reject Reason

Note: — The parameter is used to provide a reason for rejecting a DSC dialogue.

3.6.5.1 If , and only if, the CPDLC-ground-user rejects the request to open a DSC dialogue, the CPDLC-ground-user shall provide a reason (a CPDLC message) for the rejection.

3.6.5.2 The *Reject Reason* parameter shall conform to the ASN.1 abstract syntax ATCUplinkMessage.

3.6.6 Result

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

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

3.6.7 Class of Communication Service

Note: — 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.

3.6.7.1 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”, “I”, or “J”~~.~~

Note: — Class of Communication service parameter values are detailed in [5].

3.7 CPDLC-message Service

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

3.7.1 The CPDLC-message service primitives shall contain the parameters as presented Table 3-3.

Parameter Name	Req	Ind
CPDLC Message	M	M(=)

Table 3-3: CPDLC-message Service Parameters

3.7.2 CPDLC Message

Note: — This parameter contains a CPDLC message.

3.7.2.1 The CPDLC Message parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage, if provided by the CPDLC-ground-user.

3.7.2.2 The CPDLC Message parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage, if provided by the CPDLC-air-user.

3.8 CPDLC-end Service

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

3.8.1 The CPDLC-end service primitives shall contain the parameters as presented Table 3-4.

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

Table 3-4: CPDLC-end Service Parameters

3.8.2   CPDLC Message

Note: — This parameter contains a CPDLC message.

3.8.2.1   The CPDLC Message parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage, if provided by the CPDLC-ground-user.

3.8.2.2   The CPDLC Message parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage, if provided by the CPDLC-air-user.

3.8.3   Result

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

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

3.9   DSC-end Service

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

3.9.1   The DSC-end service primitives shall contain the parameters as presented Table 3-5.

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

Table 3-5: DSC-end Service Parameters

3.9.2   CPDLC Message

Note: — This parameter contains a CPDLC message.

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

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

3.9.3   Result

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

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

3.10   CPDLC-forward Service

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

3.10.1 The CPDLC-forward service primitives shall contain the parameters as presented Table 3-6.

Parameter Name	Req	Ind	Cnf
Called ICAO Facility Designator	M		
Calling ICAO Facility Designator	M	M(=)	
CPDLC Message	M	M(=)	
Class of Communication Service	U		

Table 3-6: CPDLC-forward Service Parameters

3.10.2 Called ICAO Facility Designator

Note: — This parameter contains the addressed ground system's ICAO facility designator.

3.10.2.1 The *Called ICAO Facility Designator* parameter value shall conform to the abstract syntax eight-character ICAO facility designator.

3.10.3 Calling ICAO Facility Designator

Note: — This parameter contains the sending ground system's ICAO facility designator.

3.10.3.1 The *Calling ICAO Facility Designator* parameter value shall conform to the abstract syntax eight-character ICAO facility designator.

3.10.4 CPDLC Message

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

3.10.4.1 The *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCForwardMessage, when supplied by the CPDLC-ground-user.

3.10.5 Class of Communication Service

Note: — 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.

3.10.5.1 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", "I", or "J"..

Note: — Class of Communication service parameter values are detailed in [5].

3.11 CPDLC-user-abort Service

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

3.11.1 The CPDLC-user-abort service primitives shall contain the parameters as presented Table 3-7.

.

Parameter Name	Req	Ind
Reason	U	C(=)

Table 3-7: CPDLC-user-abort Service Parameters

3.11.2 Reason

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

3.11.2.1 The *Reason* parameter value conforms to the ASN.1 abstract syntax CPDLCAbortReason.

3.12 CPDLC-provider-abort Service

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

3.12.1 The CPDLC-provider-abort service primitives shall contain the parameters as presented Table 3-8.

Parameter Name	Ind
Reason	M

Table 3-8: CPDLC-provider-abort Service Parameters

3.12.2 Reason

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

3.12.2.1 The *Reason* parameter shall conform to the ASN.1 abstract syntax CPDLCAbortReason.

4. FORMAL DEFINITIONS OF MESSAGES

4.1 CPDLC ASN.1 Abstract Syntax

4.1.1 The abstract syntax of the CPDLC protocol data units shall comply with the description contained in the ASN.1 module CPDLCMessageSetVersion1 (conforming to [3]), as defined in this section.

```
CPDLCMessageSetVersion1 DEFINITIONS~~ AUTOMATIC TAGS~~::=

BEGIN

-- -------------------------------------------------------------------------------
-- Ground Generated Messages - Top level
-- -------------------------------------------------------------------------------
GroundPDUs ::= CHOICE
        {
        abort               [0]     CPDLCAbortReason,
        startup             [1]     UplinkMessage,
        send                [2]     ATCUplinkMessage,
        forward             [3]     ATCForwardMessage,
        ...
        }

UplinkMessage ::= CHOICE
        {
        noMessage                   [0]     NULL,
        aTCUplinkMessage            [1]     ATCUplinkMessage
        }

ATCUplinkMessage ::= SEQUENCE
        {
        header                      ATCMessageHeader,
        e~~E~~lementIds                 SEQUENCE SIZE (1..5) OF ATCUplinkMsgElementId
        }

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

ForwardHeader ::= SEQUENCE
        {
        dateTime                    DateTimeGroup,
        aircraftID                  AircraftFlightIdentification,
        airframeID                  AirFrameID
        }

ForwardMessage ::= CHOICE
        {
        upElementIDs                [0]     SEQUENCE SIZE (1..5) OF ATCUplinkMsgElementId,
        downElementIDs              [1]     SEQUENCE SIZE (1..5) OF ATCDownlinkMsgElementId
        }

-- ----------------------------------------------------------------------------------
-- Aircraft Generated Messages - Top level
-- ----------------------------------------------------------------------------------
AircraftPDUs::= CHOICE
        {
        abort               [0]     CPDLCAbortReason,
        startdown           [1]     StartDownMessage,
        send                [2]     ATCDownlinkMessage,
        ...
        }

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

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

DownlinkMessage ::=   CHOICE
        {
        noMessage                          [0]     NULL,
        aTCDownlinkMessage                 [1]     ATCDownlinkMessage
        }

ATCDownlinkMessage ::=   SEQUENCE
        {
        header                  ATCMessageHeader,
        e~~E~~lementIds             SEQUENCE SIZE (1..5) OF ATCDownlinkMsgElementId
        }

-- ----------------------------------------------------------------------------------
-- 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)
```

```
CPDLCAbortReason ::= ENUMERATED
        {
        c~~C~~PDLC-user-abort                          (0),
        no-message-identification-numbers-available    (1),
        duplicate-message-identification-numbers       (2),
        no-longer-next-data-authority                  (3),
        current-data-authority-abort                   (4),
        timer-expired                                  (5),
        undefined-error                                (6),
        invalid-PDU                                    (7),
        not-permitted-PDU                              (8),
        communication-service-error                    (9),
        communication-service-failure                  (10)~~,~~
        ...
        }

LogicalAck ::= ENUMERATED
        {
        required                    (0),
        notRequired                 (1)
        }

-- ------------------------------------------------------------------------------
-- Uplink message element
-- ------------------------------------------------------------------------------

ATCUplinkMsgElementId ::= CHOICE
        {
--      UNABLE                                         Urg(N)/Alr(L)/Resp(N )
        uM0NULL                                        [0] NULL,

--      STANDBY                                        Urg(N)/Alr(L)/Resp(N )
        uM1NULL                                        [1] NULL,

--      REQUEST DEFERRED                               Urg(N)/Alr(L)/Resp(N )
        uM2NULL                                        [2] NULL,

--      ROGER                                          Urg(N)/Alr(L)/Resp(N )
        uM3NULL                                        [3] NULL,

--      AFFIRM                                         Urg(N)/Alr(L)/Resp(N )
        uM4NULL                                        [4] NULL,

--      NEGATIVE                                       Urg(N)/Alr(L)/Resp(N )
        uM5NULL                                        [5] NULL,

--      EXPECT [altitude]                              Urg(L)/Alr(L)/Resp( R )
        uM6Altitude                                    [6] Altitude,

--      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 [altitude]     Urg(L)/Alr(L)/Resp( R )
        uM13TimeAltitude                         [13] TimeAltitude,

--      AT [position] EXPECT CLIMB TO [altitude] Urg(L)/Alr(L)/Resp( R )
        uM14PositionAltitude                     [14] PositionAltitude,

--      AT [time] EXPECT DESCENT TO [altitude]   Urg(L)/Alr(L)/Resp( R )
        uM15TimeAltitude                         [15] TimeAltitude,

--      AT [position] EXPECT DESCENT TO [altitude]  Urg(L)/Alr(L)/Resp( R )
        uM16PositionAltitude                     [16] PositionAltitude,

--      AT [time] EXPECT CRUISE CLIMB TO [altitude]  Urg(L)/Alr(L)/Resp( R )
        uM17TimeAltitude                         [17] TimeAltitude,

--      AT [position] EXPECT CRUISE CLIMB TO [altitude]
                                                 Urg(L)/Alr(L)/Resp( R )
        uM18PositionAltitude                     [18] PositionAltitude,

--      MAINTAIN [altitude]                      Urg(N)/Alr(M)/Resp(W/U)
        uM19Altitude                             [19] Altitude,

--      CLIMB TO AND MAINTAIN [altitude]         Urg(N)/Alr(M)/Resp(W/U)
        uM20Altitude                             [20] Altitude,

--      AT [time] CLIMB TO AND MAINTAIN [altitude]  Urg(N)/Alr(M)/Resp(W/U)
        uM21TimeAltitude                         [21] TimeAltitude,

--      AT [position] CLIMB TO AND MAINTAIN [altitude]
                                                 Urg(N)/Alr(M)/Resp(W/U)
        uM22PositionAltitude                     [22] PositionAltitude,

--      DESCEND TO AND MAINTAIN [altitude]       Urg(N)/Alr(M)/Resp(W/U)
        uM23Altitude                             [23] Altitude,

--      AT [time] DESCEND TO AND MAINTAIN [altitude]
                                                 Urg(N)/Alr(M)/Resp(W/U)
        uM24TimeAltitude                         [24] TimeAltitude,
--      AT [position] DESCEND TO AND MAINTAIN [altitude]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM25PositionAltitude                    [25] PositionAltitude,

--      CLIMB TO REACH [altitude] BY [time]     Urg(N)/Alr(M)/Resp(W/U)
        uM26AltitudeTime                        [26] AltitudeTime,

--      CLIMB TO REACH [altitude] BY [position] Urg(N)/Alr(M)/Resp(W/U)
        uM27AltitudePosition                    [27] AltitudePosition,

--      DESCEND TO REACH [altitude] BY [time]   Urg(N)/Alr(M)/Resp(W/U)
        uM28AltitudeTime                        [28] AltitudeTime,

--      DESCEND TO REACH [altitude] BY [position] Urg(N)/Alr(M)/Resp(W/U)
        uM29AltitudePosition                    [29] AltitudePosition,

--      MAINTAIN BLOCK [altitude] TO [altitude] Urg(N)/Alr(M)/Resp(W/U)
        uM30AltitudeAltitude                    [30] AltitudeAltitude,

--      CLIMB TO AND MAINTAIN BLOCK [altitude] TO [altitude]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM31AltitudeAltitude                    [31] AltitudeAltitude,

--      DESCEND TO AND MAINTAIN BLOCK [altitude] TO [altitude]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM32AltitudeAltitude                    [32] AltitudeAltitude,

--      CLEARED OUT OF CONTROLLED AIRSPACE      Urg(N)/Alr(M)/Resp(W/U)
        uM33NULL                                [33] NULL,

--      CRUISE CLIMB TO [altitude]              Urg(N)/Alr(M)/Resp(W/U)
        uM34Altitude                            [34] Altitude,

--      CRUISE CLIMB ABOVE [altitude]           Urg(N)/Alr(M)/Resp(W/U)
        uM35Altitude                            [35] Altitude,

--      EXPEDITE CLIMB TO [altitude]            Urg(U)/Alr(M)/Resp(W/U)
        uM36Altitude                            [36] Altitude,

--      EXPEDITE DESCENT TO [altitude]          Urg(U)/Alr(M)/Resp(W/U)
        uM37Altitude                            [37] Altitude,

--      IMMEDIATELY CLIMB TO [altitude]         Urg(D)/Alr(H)/Resp(W/U)
        uM38Altitude                            [38] Altitude,

--      IMMEDIATELY DESCEND TO [altitude]       Urg(D)/Alr(H)/Resp(W/U)
        uM39Altitude                            [39] Altitude,

--      IMMEDIATELY STOP CLIMB AT [altitude]    Urg(D)/Alr(H)/Resp(W/U)
        uM40Altitude                            [40] Altitude,

--      IMMEDIATELY STOP DESCENT AT [altitude]  Urg(D)/Alr(H)/Resp(W/U)
        uM41Altitude                            [41] Altitude,
--      EXPECT TO CROSS [position] AT [altitude]          Urg(L)/Alr(L)/Resp( R )
        uM42PositionAltitude                              [42] PositionAltitude,

--      EXPECT TO CROSS [position] AT OR ABOVE [altitude]
                                                          Urg(L)/Alr(L)/Resp( R )
        uM43PositionAltitude                              [43] PositionAltitude,

--      EXPECT TO CROSS [position] AT OR BELOW [altitude]
                                                          Urg(L)/Alr(L)/Resp( R )
        uM44PositionAltitude                              [44] PositionAltitude,

--      EXPECT TO CROSS [position] AT AND MAINTAIN [altitude]
                                                          Urg(L)/Alr(L)/Resp( R )
        uM45PositionAltitude                              [45] PositionAltitude,

--      CROSS [position] AT [altitude]                    Urg(N)/Alr(M)/Resp(W/U)
        uM46PositionAltitude                              [46] PositionAltitude,

--      CROSS [position] AT OR ABOVE [altitude]           Urg(N)/Alr(M)/Resp(W/U)
        uM47PositionAltitude                              [47] PositionAltitude,

--      CROSS [position] AT OR BELOW [altitude]           Urg(N)/Alr(M)/Resp(W/U)
        uM48PositionAltitude                              [48]PositionAltitude,

--      CROSS [position] AT AND MAINTAIN [altitude]       Urg(N)/Alr(M)/Resp(W/U)
        uM49PositionAltitude                              [49] PositionAltitude,

--      CROSS [position] BETWEEN [altitude] AND [altitude]
                                                          Urg(N)/Alr(M)/Resp(W/U)
        uM50PositionAltitudeAltitude                      [50] PositionAltitudeAltitude,

--      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 [altitude]          Urg(N)/Alr(M)/Resp(W/U)
```

```
        uM58PositionTimeAltitude                [58] PositionTimeAltitude,

--      CROSS [position] AT OR BEFORE [time] AT [altitude]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM59PositionTimeAltitude                [59] PositionTimeAltitude,

--      CROSS [position] AT OR AFTER [time] AT [altitude]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM60PositionTimeAltitude                [60] PositionTimeAltitude,

--      CROSS [position] AT AND MAINTAIN [altitude] AT [speed]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM61PositionAltitudeSpeed               [61] PositionAltitudeSpeed,

--      AT [time] CROSS [position] AT AND MAINTAIN [altitude]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM62TimePositionAltitude                [62] TimePositionAltitude,

--      AT [time] CROSS [position] AT AND MAINTAIN [altitude] AT [speed]
--                                              Urg(N)/Alr(M)/Resp(W/U)
        uM63TimePositionAltitudeSpeed           [63] TimePositionAltitudeSpeed,

--      OFFSET [distanceOffset] [direction] OF ROUTE    Urg(N)/Alr(M)/Resp(W/U)
        uM64DistanceOffsetDirection             [64] DistanceOffsetDirection,

--      AT [position] OFFSET [distanceOffset] [direction] OF ROUTE
--                                              Urg(N)/Alr(M)/Resp(W/U)
        uM65PositionDistanceOffsetDirection     [65] PositionDistanceOffsetDirection,

--      AT [time] OFFSET [distanceOffset] [direction] OF ROUTE
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM66TimeDistanceOffsetDirection         [66] TimeDistanceOffsetDirection,

--      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 [altitude] PROCEED DIRECT TO [position]         Urg(N)/Alr(M)/Resp(W/U)
        uM78AltitudePosition                               [78] AltitudePosition,

--      CLEARED TO [position] VIA [routeClearance]         Urg(N)/Alr(M)/Resp(W/U)
        uM79PositionRouteClearance                         [79] PositionRouteClearance,

--      CLEARED [routeClearance]                           Urg(N)/Alr(M)/Resp(W/U)
        uM80RouteClearance                                 [80] RouteClearance,

--      CLEARED [procedureName]                            Urg(N)/Alr(M)/Resp(W/U)
        uM81ProcedureName                                  [81] ProcedureName,

--      CLEARED TO DEVIATE UP TO [distanceOffset] [direction] OF ROUTE
                                                           Urg(N)/Alr(M)/Resp(W/U)
        uM82DistanceOffsetDirection                        [82] DistanceOffsetDirection,

--      AT [position] CLEARED [routeClearance]             Urg(N)/Alr(M)/Resp(W/U)
        uM83PositionRouteClearance                         [83] PositionRouteClearance,

--      AT [position] CLEARED [procedureName]              Urg(N)/Alr(M)/Resp(W/U)
        uM84PositionProcedureName                          [84] PositionProcedureName,

--      EXPECT [routeClearance]                            Urg(L)/Alr(L)/Resp( R )
        uM85RouteClearance                                 [85] RouteClearance,

--      AT [position] EXPECT [routeClearance]              Urg(L)/Alr(L)/Resp( R )
        uM86PositionRouteClearance                         [86] PositionRouteClearance,

--      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 [altitude] EXPECT DIRECT TO [position]          Urg(L)/Alr(L)/Resp( R )
        uM90AltitudePosition                               [90] AltitudePosition,

--      HOLD AT [position] MAINTAIN [altitude] INBOUND TRACK [degrees][direction]
--      TURNS [legtype]                                    Urg(N)/Alr(M)/Resp(W/U)
```

```
        uM91HoldClearance                           [91] HoldClearance,

--      HOLD AT [position] AS PUBLISHED MAINTAIN [altitude]
                                                    Urg(N)/Alr(M)/Resp(W/U)
        uM92PositionAltitude                        [92] PositionAltitude,

--      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,

--      FLY 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 [altitude] EXPECT [speed]                Urg(L)/Alr(L)/Resp(R)
        uM102AltitudeSpeed                          [102] AltitudeSpeed,

--      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 [altitude] EXPECT [speed] TO [speed]     Urg(L)/Alr(L)/Resp(R)
        uM105AltitudeSpeedSpeed                     [105] AltitudeSpeedSpeed,

--      MAINTAIN [speed]                            Urg(N)/Alr(M)/Resp(W/U)
        uM106Speed                                  [106] Speed,

--      MAINTAIN PRESENT SPEED                      Urg(N)/Alr(M)/Resp(W/U)
        uM107NULL                                   [107] NULL,

--      MAINTAIN [speed] OR GREATER                 Urg(N)/Alr(M)/Resp(W/U)
```

```
        uM108Speed                              [108] Speed,

--      MAINTAIN [speed] OR LESS                Urg(N)/Alr(M)/Resp(W/U)
        uM109Speed                              [109] Speed,

--      MAINTAIN [speed] TO [speed]             Urg(N)/Alr(M)/Resp(W/U)
        uM110SpeedSpeed                         [110] SpeedSpeed,

--      INCREASE SPEED TO [speed]               Urg(N)/Alr(M)/Resp(W/U)
        uM111Speed                              [111] Speed,

--      INCREASE SPEED TO [speed] OR GREATER    Urg(N)/Alr(M)/Resp(W/U)
        uM112Speed                              [112] Speed,

--      REDUCE SPEED TO [speed]                 Urg(N)/Alr(M)/Resp(W/U)
        uM113Speed                              [113] Speed,

--      REDUCE SPEED TO [speed] OR LESS         Urg(N)/Alr(M)/Resp(W/U)
        uM114Speed                              [114] Speed,

--      DO NOT EXCEED [speed]                   Urg(N)/Alr(M)/Resp(W/U)
        uM115Speed                              [115] Speed,

--      RESUME NORMAL SPEED                     Urg(N)/Alr(M)/Resp(W/U)
        uM116NULL                               [116] NULL,

--      CONTACT [icaounitname] [frequency]      Urg(N)/Alr(M)/Resp(W/U)
        uM117ICAOUnitNameFrequency              [117] ICAOUnitNameFrequency,

--      AT [position] CONTACT [icaounitname] [frequency]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM118PositionICAOUnitNameFrequency      [118] PositionICAOUnitNameFrequency,

--      AT [time] CONTACT [icaounitname] [frequency]   Urg(N)/Alr(M)/Resp(W/U)
        uM119TimeICAOUnitNameFrequency          [119] TimeICAOUnitNameFrequency,

--      MONITOR [icaounitname] [frequency]      Urg(N)/Alr(M)/Resp(W/U)
        uM120ICAOUnitNameFrequency              [120] ICAOUnitNameFrequency,

--      AT [position] MONITOR [icaounitname] [frequency]
                                                Urg(N)/Alr(M)/Resp(W/U)
        uM121PositionICAOUnitNameFrequency      [121] PositionICAOUnitNameFrequency,

--      AT [time] MONITOR [icaounitname] [frequency]   Urg(N)/Alr(M)/Resp(W/U)
        uM122TimeICAOUnitNameFrequency          [122] TimeICAOUnitNameFrequency,

--      SQUAWK [beaconcode]                     Urg(N)/Alr(M)/Resp(W/U)
        uM123BeaconCode                         [123] BeaconCode,

--      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(M)/Resp( R )
        uM127NULL                                   [127] NULL,

--      REPORT LEAVING [altitude]                   Urg(N)/Alr(M)/Resp( R )
        uM128Altitude                               [128] Altitude,

--      REPORT WHEN LEVEL AT [altitude]             Urg(N)/Alr(M)/Resp(R)
        uM129Altitude                               [129] Altitude,

--      REPORT PASSING [position]                   Urg(N)/Alr(M)/Resp( R )
        uM130Position                               [130] Position,

--      REPORT REMAINING FUEL AND PERSONS~~SOULS~~ ON BOARD
                                                         Urg(N)/Alr(M)/Resp(Y )
        uM131NULL                                   [131] NULL,

--      REPORT POSITION                             Urg(N)/Alr(M)/Resp(Y)
        uM132NULL                                   [132] NULL,

--      REPORT ALTITUDE                             Urg(N)/Alr(M)/Resp(Y)
        uM133NULL                                   [133] NULL,

--      REPORT SPEED [speedqualifier] [speedtype]   Urg(N)/Alr(M)/Resp(Y)
        uM134-SpeedQualifierSpeedType               [134] SpeedQualifierSpeedType,

--      CONFIRM ASSIGNED LEVEL                      Urg(N)/Alr(M)/Resp(Y )
        uM135NULL                                   [135] NULL,

--      CONFIRM ASSIGNED SPEED                      Urg(N)/Alr(M)/Resp(Y )
        uM136NULL                                   [136] NULL,

--      CONFIRM ASSIGNED ROUTE                      Urg(N)/Alr(M)/Resp(Y )
        uM137NULL                                   [137] NULL,

--      CONFIRM TIME OVER REPORTED WAYPOINT Urg(N)/Alr(M)/Resp(Y)
        uM138NULL                                   [138] NULL,

--      CONFIRM REPORTED WAYPOINT                   Urg(N)/Alr(M)/Resp(Y)
        uM139NULL                                   [139] NULL,

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

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

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

--      CONFIRM REQUEST                             Urg(N)/Alr(M)/Resp(Y)
```

```
        uM143NULL                                   [143] NULL,

--      CONFIRM SQUAWK                              Urg(N)/Alr(M)/Resp(Y)
        uM144NULL                                   [144] NULL,

--      REPORT HEADING                              Urg(N)/Alr(M)/Resp(Y)
        uM145NULL                                   [145] NULL,

--      REPORT GROUND TRACK                         Urg(N)/Alr(M)/Resp(Y)
        uM146NULL                                   [146] NULL,

--      REQUEST POSITION REPORT                     Urg(N)/Alr(M)/Resp(Y )
        uM147NULL                                   [147] NULL,

--      WHEN CAN YOU ACCEPT [altitude]              Urg(N)/Alr(M)/Resp(Y)
        uM148Altitude                               [148] Altitude,

--      CAN YOU ACCEPT [altitude] AT [position]     Urg(N)/Alr(M)/Resp(A/N)
        uM149AltitudePosition                       [149] AltitudePosition,

--      CAN YOU ACCEPT [altitude] AT [time]         Urg(N)/Alr(M)/Resp(A/N)
        uM150AltitudeTime                           [150] AltitudeTime,

--      WHEN CAN YOU ACCEPT [speed]                 Urg(N)/Alr(M)/Resp(Y)
        uM151Speed                                  [151] Speed,

--      WHEN CAN YOU ACCEPT [distanceOffset] [direction] OFFSET
                                                    Urg(N)/Alr(M)/Resp(Y)
        uM152DistanceOffsetDirection                [152] DistanceOffsetDirection,

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

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

--      RADAR CONTACT [position]                    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(M)/Resp(R)
        uM158AtisCode                               [158] ATIS~~tis~~Code,

--      ERROR [errorInformation]                    Urg(U)/Alr(M)/Resp(N)
        uM159ErrorInformation                       [159] ErrorInformation,

--      NEXT DATA AUTHORITY [icaofacilitydesignator] Urg(L)/Alr(N)/Resp(N)
        uM160ICAOFacilityDesignator                 [160] ICAOFacilityDesignator,
--      END SERVICE                              Urg(L)/Alr(N)/Resp(N)
        uM161NULL                                [161] NULL,

--      SERVICE UNAVAILABLE                      Urg(L)/Alr(L)/Resp(N )
        uM162NULL                                [162] NULL,

--      [icaofacilitydesignator]                 Urg(L)/Alr(N)/Resp(N)
        uM163ICAOFacilityDesignator              [163] ICAOFacilityDesignator,

--      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(N)/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 [altitude]               Urg(N)/Alr(M)/Resp(R)
        uM175Altitude                            [175] Altitude,

--      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,

--      [freetext]                               Urg(N)/Alr(L)/Resp(N)
        uM178FreeText                            [178] FreeText,
--      SQUAWK IDENT                                    Urg(N)/Alr(M)/Resp(W/U)
        uM179NULL                                       [179] NULL,

--      REPORT REACHING BLOCK [altitude] TO [altitude]
                                                        Urg(N)/Alr(M)/Resp(R)
        uM180AltitudeAltitude                           [180] AltitudeAltitude,

--      REPORT DISTANCE [tofrom] [position]             Urg(N)/Alr(M)/Resp(Y)
        uM181ToFromPosition                             [181] ToFromPosition,

--      CONFIRM ATIS CODE                               Urg(N)/Alr(M)/Resp(Y)
        uM182NULL                                       [182] NULL,

--      [freetext]                                      Urg(N)/Alr(M)/Resp(N)
        uM183FreeText                                   [183] FreeText,

--      AT [time] REPORT DISTANCE [tofrom] [position]   Urg(N)/Alr(M)/Resp(Y)
        uM184TimeToFromPosition                         [184] TimeToFromPosition,

--      AFTER PASSING [position] CLIMB TO AND MAINTAIN [altitude]
                                                        Urg(N)/Alr(M)/Resp(W/U)
        uM185PositionAltitude                           [185] PositionAltitude,

--      AFTER PASSING [position] DESCEND TO AND MAINTAIN [altitude]
                                                        Urg(N)/Alr(M)/Resp(W/U)
        uM186PositionAltitude                           [186] PositionAltitude,

--      [freetext]                                      Urg(L)/Alr(N)/Resp(N)
        uM187FreeText                                   [187] FreeText,

--      AFTER PASSING [position] MAINTAIN [speed]       Urg(N)/Alr(M)/Resp(W/U)
        uM188PositionSpeed                              [188] PositionSpeed,

--      ADJUST SPEED TO [speed]                         Urg(N)/Alr(M)/Resp(W/U)
        uM189Speed                                      [189] NULL,

--      FLY HEADING [degrees]                           Urg(N)/Alr(M)/Resp(W/U)
        uM190Degrees                                    [190] Degrees,

--      ALL ATS TERMINATED                              Urg(N)/Alr(M)/Resp(R)
        uM191NULL                                       [191] NULL,

--      REACH [altitude] BY [time]                      Urg(N)/Alr(M)/Resp(W/U)
        uM192AltitudeTime                               [192] AltitudeTime,

--      IDENTIFICATION LOST                             Urg(N)/Alr(M)/Resp(R)
        uM193NULL                                       [193] NULL,

--      [freetext]                                      Urg(N)/Alr(L)/Resp(Y)
        uM194FreeText                                   [194] FreeText,

--      [freetext]                                      Urg(L)/Alr(L)/Resp(R)
        uM195FreeText                                   [195] FreeText,
--    [freetext]                                           Urg(N)/Alr(VA/Resp(W/U)
      uM196FreeText                                        [196] FreeText,

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

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

--    [freetext]                                           Urg(N)/Alr(M)/Resp(W/U)
      uM199FreeText                                        [199] FreeText,

--    [freetext]                                           Urg(L)/Alr(L)/Resp(R)
      uM200FreeText                                        [200] FreeText,

--    [freetext]                                           Urg(N)/Alr(M)/Resp(W/U)
      uM201FreeText                                        [201] FreeText,

--    [freetext]                                           Urg(D)/Alr(H)/Resp(W/U)
      uM202FreeText                                        [202] FreeText,

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

--    [freetext]                                           Urg(N)/Alr(M)/Resp(Y)
      uM204FreeText                                        [204] FreeText,

--    [freetext]                                           Urg(N)/Alr(M)/Resp(A/N)
      uM205FreeText                                        [205] FreeText,

--    [freetext]                                           Urg(L)/Alr(N)/Resp(Y)
      uM206FreeText                                        [206] FreeText,

--    [freetext]                                           Urg(L)/Alr(L)/Resp(Y)
      uM207FreeText                                        [207] FreeText,

--    [freetext]                                           Urg(L)/Alr(L)/Resp(N)
      uM208FreeText                                        [208] FreeText,

--    REACH [altitude] BY [position]                       Urg(N)/Alr(M)/Resp(W/U)
      uM209AltitudePosition                                [209] AltitudePosition,

--    IDENTIFIED [position]                                Urg(N)/Alr(M)/Resp(R)
      uM210Position                                        [210] Position,

--    REQUEST FORWARDED                                    Urg(N)/Alr(L)/Resp(N)
      uM211NULL                                            [211] NULL,

--    [icaofacilitydesignator] ATIS [atiscode] CURRENT     Urg(N)/Alr(M)/Resp(R)
      uM212ICAOFacilityDesignatorATISCode                  [212] ICAOFacilityDesignatorATISCode,

--    [icaofacilitydesignator] ALTIMETER [altimeter]       Urg(N)/Alr(M)/Resp(R)
      uM213ICAOFacilityDesignatorAltimeter                 [213] ICAOFacilityDesignatorAltimeter,
--      RUNWAY [runway] VISUAL RANGE [rvr]        Urg(N)/Alr(M)/Resp(R)
        uM214RunwayRVR                            [214] RunwayRVR,

--      TURN [degrees][direction]                 Urg(N)/Alr(M)/Resp(W/U)
        uM215DegreesDirection                     [215] ~~Degrees~~DirectionDegrees,

--      REQUEST FLIGHT PLAN                       Urg(N)/Alr(M)/Resp(Y)
        uM216NULL                                 [216] NULL,

--      REPORT ARRIVAL                            Urg(N)/Alr(M)/Resp(Y)
        uM217NULL                                 [217] NULL,

--      REQUEST ALREADY RECEIVED                  Urg(L)/Alr(N)/Resp(N)
        uM218NULL                                 [218] NULL,

--      STOP CLIMB AT [altitude]                  Urg(U)/Alr(M)/Resp(W/U)
        uM219Altitude                             [219] Altitude,

--      STOP DESCENT AT [altitude]                Urg(U)/Alr(M)/Resp(W/U)
        uM220Altitude                             [220] Altitude,

--      STOPTURN HEADING [degrees]                Urg(U)/Alr(M)/Resp(W/U)
        uM221Degrees                              [221] Degrees,

--      NO SPEED RESTRICTION                      Urg(L)/Alr(L)/Resp(R)
        uM222NULL                                 [222] NULL,
~~,~~
--      REDUCE TO MINIMUM APPROACH SPEED          Urg(N)/Alr(M~~L~~)/Resp(W/U)
        uM223NULL                                 [223] NULL,

--      NO DELAY EXPECTED                         Urg(N)/Alr(L)/Resp(R)
        uM224NULL                                 [224] NULL,

--      DELAY NOT DETERMINED                      Urg(N)/Alr(L)/Resp(R)
        uM225NULL                                 [225] NULL,

--      EXPECTED APPROACH TIME [time]             Urg(N)/Alr(L)/Resp(R)
        uM226Time                                 [226] Time,

--      LOGICAL ACKNOWLEDGMENT                    Urg(N)/Alr(M)/Resp(N)
        uM227NULL                                 [227] NULL,

--      REPORT ETA [position]                     Urg(L)/Alr(L)/Resp(Y)
        uM228-Position                            [228] Position,

--      REPORT ALTERNATE AERODROME                Urg(L)/Alr(L)/Resp(Y)
        uM229NULL                                 [229] NULL,

--      IMMEDIATELY                               Urg(D)/Alr(H)/Resp(N)
        uM230NULL                                 [230] NULL,

--      STATE PREFERRED ALTITUDE                  Urg(L)/Alr(L)/Resp(Y)
        uM231NULL                                 [231] NULL,
--      STATE-TOP-OF-DESCENT                            Urg(L)/Alr(L)/Resp(Y)
        uM232NULL                                       [232] NULL,

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

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

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

--      CRUISE [altitude]                               Urg(N)/Alr(M)/Resp(W/U)
        uM236Altitude                                   [236] Altitude,



        ...
        }

-- --------------------------------------------------------------------------------
-- 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 [altitude]                              Urg(N)/Alr(L)/Resp(Y)
        dM6Altitude                                     [6] Altitude,

--      REQUEST BLOCK [altitude] TO [altitude]          Urg(N)/Alr(L)/Resp(Y)
        dM7AltitudeAltitude                             [7] AltitudeAltitude,

--      REQUEST CRUISE CLIMB TO [altitude]              Urg(N)/Alr(L)/Resp(Y)
        dM8Altitude                                     [8] Altitude,
--      REQUEST CLIMB TO [altitude]                     Urg(N)/Alr(L)/Resp(Y)
        dM9Altitude                                     [9] Altitude,

--      REQUEST DESCENT TO [altitude]                   Urg(N)/Alr(L)/Resp(Y)
        dM10Altitude                                    [10] Altitude,

--      AT [position] REQUEST CLIMB TO [altitude]       Urg(N)/Alr(L)/Resp(Y)
        dM11PositionAltitude                            [11] PositionAltitude,

--      AT [position] REQUEST DESCENT TO [altitude]     Urg(N)/Alr(L)/Resp(Y)
        dM12PositionAltitude                            [12] PositionAltitude,

--      AT [time] REQUEST CLIMB TO [altitude]           Urg(N)/Alr(L)/Resp(Y)
        dM13TimeAltitude                                [13] TimeAltitude,

--      AT [time] REQUEST DESCENT TO [altitude]         Urg(N)/Alr(L)/Resp(Y)
        dM14TimeAltitude                                [14] TimeAltitude,

--      REQUEST OFFSET [distanceOffset] [direction] OF ROUTE
                                                        Urg(N)/Alr(L)/Resp(Y)
        dM15DistanceOffsetDirection                     [15] DistanceOffsetDirection,

--      AT [position] REQUEST OFFSET [distanceOffset] [direction] OF ROUTE
                                                        Urg(N)/Alr(L)/Resp(Y)
        dM16PositionDistanceOffsetDirection             [16] PositionDistanceOffsetDirection,

--      AT [time] REQUEST OFFSET [distanceOffset] [direction] OF ROUTE
                                                        Urg(N)/Alr(L)/Resp(Y)
        dM17TimeDistanceOffsetDirection                 [17] TimeDistanceOffsetDirection,

--      REQUEST [speed]                                 Urg(N)/Alr(L)/Resp(Y)
        dM18Speed                                       [18] Speed,

--      REQUEST [speed] TO [speed]                      Urg(N)/Alr(L)/Resp(Y)
        dM19SpeedSpeed                                  [19] SpeedSpeed,

--      REQUEST VOICE CONTACT                           Urg(N)/Alr(L)/Resp(Y)
        dM20NULL                                        [20] NULL,

--      REQUEST VOICE CONTACT [frequency]               Urg(N)/Alr(L)/Resp(Y)
        dM21Frequency                                   [21] Frequency,

--      REQUEST DIRECT TO [position]                    Urg(N)/Alr(L)/Resp(Y)
        dM22Position                                    [22] Position,

--      REQUEST [procedureName]                         Urg(N)/Alr(L)/Resp(Y)
        dM23ProcedureName                               [23] ProcedureName,

--      REQUEST [routeClearance]                        Urg(N)/Alr(L)/Resp(Y)
        dM24RouteClearance                              [24] RouteClearance,

--      REQUEST [clearanceType] CLEARANCE               Urg(N)/Alr(L)/Resp(Y)
        dM25ClearanceType                               [25] ClearanceType,
--      REQUEST WEATHER DEVIATION TO  [position] VIA [routeClearance]
                                                Urg(N)/Alr(L)/Resp(Y)
        dM26PositionRouteClearance              [26] PositionRouteClearance,

--      REQUEST WEATHER DEVIATION UP  TO [distanceOffset] [direction] OF ROUTE
                                                Urg(N)/Alr(L)/Resp(Y)
        dM27DistanceOffsetDirection             [27] DistanceOffsetDirection,

--      LEAVING [altitude]                      Urg(N)/Alr(L)/Resp(Y)
        dM28Altitude                            [28] Altitude,

--      CLIMBING TO [altitude]                  Urg(N)/Alr(M)/Resp(N)
        dM29Altitude                            [29]Altitude,

--      DESCENDING TO [altitude]                Urg(N)/Alr(M)/Resp(N)
        dM30Altitude                            [30] Altitude,

--      PASSING [position]                      Urg(N)/Alr(M)/Resp(N)
        dM31Position                            [31] Position,

--      PRESENT ALTITUDE [altitude]             Urg(N)/Alr(M)/Resp(N)
        dM32Altitude                            [32] Altitude,

--      PRESENT POSITION [position]             Urg(N)/Alr(M)/Resp(N)
        dM33Position                            [33] Position,

--      PRESENT SPEED [speed]                   Urg(N)/Alr(M)/Resp(N)
        dM34Speed                               [34]Speed,

--      PRESENT HEADING [degrees]               Urg(N)/Alr(M)/Resp(N)
        dM35Degrees                             [35] Degrees,

--      PRESENT GROUND TRACK [degrees]          Urg(N)/Alr(M)/Resp(N)
        dM36Degrees                             [36]Degrees,

--      LEVEL [altitude]                        Urg(N)/Alr(M)/Resp(N)
        dM37Altitude                            [37] Altitude,

--      ASSIGNED ALTITUDE [altitude]            Urg(N)/Alr(M)/Resp(N)
        dM38Altitude                            [38] Altitude,

--      ASSIGNED SPEED [speed]                  Urg(N)/Alr(M)/Resp(N)
        dM39Speed                               [39] Speed,

--      ASSIGNED ROUTE [routeClearance]         Urg(N)/Alr(M)/Resp(N)
        dM40RouteClearance                      [40] RouteClearance,

--      BACK ON ROUTE                           Urg(N)/Alr(M)/Resp(N)
        dM41NULL                                [41] NULL,

--      NEXT WAYPOINT [position]                Urg(N)/Alr(M)/Resp(N)
        dM42Position                            [42] Position,
--      NEXT WAYPOINT ETA [time]                    Urg(N)/Alr(M)/Resp(N)
        dM43Time                                    [43] Time,

--      ENSUING WAYPOINT [position]                 Urg(N)/Alr(M)/Resp(N)
        dM44Position                                [44] Position,

--      REPORTED WAYPOINT [position]                Urg(N)/Alr(M)/Resp(N)
        dM45Position                                [45] Position,

--      REPORTED WAYPOINT [time]                    Urg(N)/Alr(M)/Resp(N)
        dM46Time                                    [46] Time,

--      SQUAWKING [beaconcode]                      Urg(N)/Alr(M)/Resp(N)
        dM47BeaconCode                              [47] BeaconCode,

--      POSITION REPORT [positionreport]            Urg(N)/Alr(M)/Resp(N)
        dM48PositionReport                          [48] PositionReport,

--      WHEN CAN WE EXPECT [speed]                  Urg(L)/Alr(L)/Resp(Y)
        dM49Speed                                   [49] Speed,

--      WHEN CAN WE EXPECT [speed] TO [speed]       Urg(L)/Alr(L)/Resp(Y)
        dM50SpeedSpeed                              [50] SpeedSpeed,

--      WHEN CAN WE EXPECT BACK ON ROUTE            Urg(L)/Alr(L)/Resp(Y)
        dM51NULL                                    [51] NULL,

--      WHEN CAN WE EXPECT LOWER ALTITUDE           Urg(L)/Alr(L)/Resp(Y)
        dM52NULL                                    [52] NULL,

--      WHEN CAN WE EXPECT HIGHER ALTITUDE          Urg(L)/Alr(L)/Resp(Y)
        dM53NULL                                    [53] NULL,

--      WHEN CAN WE EXPECT CRUISE CLIMB TO [altitude]
                                                    Urg(L)/Alr(L)/Resp(Y)
        dM54Altitude                                [54] Altitude,

--      PAN PAN PAN                                 Urg(U)/Alr(H)/Resp(N)
        dM55NULL                                    [55] NULL,

--      MAYDAY MAYDAY MAYDAY                        Urg(D)/Alr(H)/Resp(N)
        dM56NULL                                    [56] NULL,

--      [remainingFuel] OF FUEL REMAINING AND [persons~~souls~~onboard] PERSONS~~SOULS~~ ON BOARD
                                                                  Urg(U)/Alr(H)/Resp(N)
        dM57RemainingFuelPersons~~Souls~~OnBoard    [57] RemainingFuelPersons~~Souls~~OnBoard,

--      CANCEL EMERGENCY                            Urg(U)/Alr(M)/Resp(N)
        dM58NULL                                    [58] NULL,

--      DIVERTING TO [position] VIA [routeClearance]  Urg(U)/Alr(H)/Resp(N)
        dM59PositionRouteClearance                  [59] PositionRouteClearance,
--      OFFSETTING [distanceOffset] [direction] OF ROUTE
                                                    Urg(U)/Alr(H)/Resp(N)
        dM60DistanceOffsetDirection                 [60] DistanceOffsetDirection,

--      DESCENDING TO [altitude]                    Urg(U)/Alr(H)/Resp(N)
        dM61Altitude                                [61] Altitude,

--      ERROR [errorInformation]                    Urg(U)/Alr(L)/Resp(N)
        dM62ErrorInformation                        [62] ErrorInformation,

--      NOT CURRENT DATA AUTHORITY                  Urg(L)/Alr(L)/Resp(N)
        dM63NULL                                    [63] NULL,

--      [icaofacilitydesignator]                    Urg(L)/Alr(L)/Resp(N)
        dM64ICAOFacilityDesignator                  [64] ICAOFacilityDesignator,

--      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 [altitude]                         Urg(N)/Alr(M)/Resp(N)
        dM72Altitude                                [72] Altitude,

--      [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 [altitude] TO [altitude]     Urg(N)/Alr(N)/Resp(N)
        dM76AltitudeAltitude                        [76] AltitudeAltitude,

--      ASSIGNED BLOCK [altitude] TO [altitude]     Urg(N)/Alr(N)/Resp(N)
```

```
        dM77AltitudeAltitude                      [77] AltitudeAltitude,

--      AT [time] [distance] [tofrom] [position]  Urg(N)/Alr(N)/Resp(N)
        dM78TimeDistanceToFromPosition            [78] TimeDistanceToFromPosition,

--      ATIS [atiscode]                           Urg(N)/Alr(M)/Resp(N)
        dM79AtisCode                              [79] ATIS~~tis~~Code,

--      DEVIATING [distanceOffset] [direction] OF ROUTE
                                                  Urg(N)/Alr(M)/Resp(N)
        dM80DistanceOffsetDirection               [80] DistanceOffsetDirection,

--      WE CAN ACCEPT [altitude] AT [time]        Urg(N)/Alr(L)/Resp(N)
        dM81AltitudeTime                          [81] AltitudeTime,

--      WE CANNNOT ACCEPT [altitude]              Urg(N)/Alr(L)/Resp(N)
        dM82Altitude                              [82] Altitude,

--      WE CAN ACCEPT [speed] AT [time]           Urg(N)/Alr(L)/Resp(N)
        dM83SpeedTime                             [83] SpeedTime,

--      WE CANNNOT ACCEPT [speed]                 Urg(N)/Alr(L)/Resp(N)
        dM84Speed                                 [84] Speed,

--      WE CAN ACCEPT [direction] [distanceOffset]  Urg(N)/Alr(L)/Resp(N)
        dM85DirectionDistanceOffset               [85] DistanceOffsetDirection,

--      WE CANNNOT ACCEPT [direction] [distanceOffset]
                                                  Urg(N)/Alr(L)/Resp(N)
        dM86DirectionDistanceOffset               [86] DistanceOffsetDirection,

--      WHEN CAN WE EXPECT CLIMB TO [altitude]    Urg(L)/Alr(L)/Resp(Y)
        dM87Altitude                              [87] Altitude,

--      WHEN CAN WE EXPECT DESCENT TO [altitude]
                                                  Urg(L)/Alr(L)/Resp(Y)
        dM88Altitude                              [88] Altitude,

--      MONITORING[icaounitname] [frequency]      Urg(L)/Alr(M)/Resp(N)
        dM89IcaounitnameFrequency                 [89] ICAO~~Icao~~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(N)/Resp(Y)
        dM103NULL                                  [103] NULL,

--      ETA[position][time]                        Urg(L)/Alr(L)/Resp(N)
        dM104PositionTime                          [104] Position-Time,

--      ALTERNATE AERODROME[airport]               Urg(L)/Alr(L)/Resp(N)
        dM105Airport                               [105] Airport,

--      PREFERRED ALTITUDE ~~LEVEL~~[altitude]                Urg(L)/Alr(L)/Resp(N)
        dM106Altitude                              [106] Altitude,

--      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,
...
}

AircraftEquipmentCode ::= SEQUENCE
{
approachEquipmentAvailable [0] BOOLEAN,
approachEquipmentStatus [1] SEQUENCE SIZE (1..24~~16~~) OF COMNAVEquipmentStatus

OPTIONAL,
sSREquipmentAvailable [2] SSREquipmentAvailable
}

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

AircraftType ::= IA5String (SIZE (2..5))

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

Airport ::= IA5String (SIZE (4))

AirwayIdentifier ::= IA5String (SIZE (1..5))

AirwayIntercept ::= IA5String (SIZE (1..5))

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 = Hecto Pascal, Range (750.0..1250.0), resolution = 0.1

Altitude ::= CHOICE
{
altitudeFeet~~Qnh~~ [0] AltitudeFeet~~Qnh~~,
altitude~~Qnh~~M~~m~~eters [1] Altitude~~Qnh~~M~~m~~eters,
~~altitudeQfe AltitudeQfe,~~
~~altitudeQfemeters AltitudeQfemeters,~~
~~altitudeGnssfeet AltitudeGnssfeet,~~
~~altitudeGnssmeters AltitudeGnssmeters,~~
altitudeFlightLevel [2] AltitudeFlightLevel,
altitudeFlightLevelMetric [3] AltitudeFlightLevelMetric
}

AltitudeAltitude ::= SEQUENCE SIZE (2) OF Altitude

AltitudeDepartureName ::= SEQUENCE
{
altitude Altitude,
procedureName ProcedureName
}

AltitudeFlightLevel ::= INTEGER (30..700~~600~~)
--unit = Level (100 Feet), Range (030..700~~600~~), resolution = 1

AltitudeFlightLevelMetric ::= INTEGER (100..2500~~2000~~)
--unit = Level (10 Meters), Range (100..2500~~.2000~~), resolution = 1

~~**AltitudeGnssfeet** ::= INTEGER (0..150000)~~
~~--unit = Foot, Range (0..150000), resolution = 1~~

~~**AltitudeGnssmeters** ::= INTEGER (0..50000)~~
~~--unit = Meter, Range (0..50000), resolution = 1~~

AltitudePosition ::= SEQUENCE
{
altitude Altitude,
position Position
}

AltitudeProcedureName ::= SEQUENCE
{
altitude Altitude,
procedureName ProcedureName
}

AltitudeFeet~~Qfe~~ ::= INTEGER (-600..70000)~~0..2100)~~
--unit = Feet, Range (-600..70000~~00..21000~~), resolution = 10

Altitude~~Qfe~~M~~m~~eters ::= INTEGER (-30..25000~~0..7000~~)
--unit = Meter, Range (-30..25000~~0..7000~~), resolution = 1

~~**AltitudeQnh** ::= INTEGER (0..2500)~~
~~--unit = Feet, Range (0..25000), resolution = 10~~

~~**AltitudeQnhmeters** ::= INTEGER (0..16000)~~
~~--unit = Meter, Range (0..16000), resolution = 1~~

AltitudeSpeed ::= SEQUENCE
{
altitude Altitude,
speed Speed
}

AltitudeSpeedSpeed ::= SEQUENCE
{
altitude Altitude,
speeds SpeedSpeed
}

```
AltitudeTime ::= SEQUENCE
        {
        altitude                Altitude,
        time                    Time
        }

ATISCode ::= IA5String (SIZE (1))

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

ATWAltitude ::= SEQUENCE
        {
        atw             ATWAltitudeTolerance,
        altitude        Altitude
        }

ATWAltitudeSequence ::= SEQUENCE SIZE (1..2) OF ATWAltitude

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

ATWDistance ::= SEQUENCE
        {
        ATWDistanceTolerance,
        Distance
        }

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

BeaconCode ::=  SEQUENCE SIZE (4) OF BeaconCodeOctalDigit

BeaconCodeOctalDigit ::= INTEGER (0..7)

ClearanceType ::= SEQUENCE
        {
        clearanceTypes          ClearanceTypes   OPTIONAL
        }
```

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

COMNAVEquipmentStatus ::= ENUMERATED
        {
        aloranA                         (0),
        cloranC                         (1),
        ddme                            (2),
        edecca                          (3),
        fadf                            (4),
        ggnss                           (5),
        hhfRTF                          (6),
        iinertialNavigation             (7),
        lils                            (8),
        momega                          (9),
        ovor                            (10),
        pdoppler                        (11),
        rrnavRouteEquipment             (12),
        ttacan                          (13),
        uuhfRTF                         (14),
        vvhfRTF                         (15),
        ...
        }

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 ~~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~~1..1024~~)
--      unit = Kilometer, Range (0..1000~~1..1024~~), resolution = 0.25~~1~~

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

DistanceOffset ::= CHOICE
        {
        distanceOffsetNm        [0]     DistanceOffsetNm,
        distanceOffsetKm        [1]     DistanceOffsetKm
        }

DistanceOffsetDirection ::=  SEQUENCE
        {
        distanceOffset          DistanceOffset,
        direction               Direction
        }

DistanceOffsetKm ::= INTEGER (1..500~~256~~)
--      unit = Kilometer, Range (1..500~~256~~), resolution = 1

DistanceOffsetNm ::= INTEGER (1..250~~128~~)
--      unit = Nautical Mile, Range (1..250~~128~~), resolution = 1

DistanceToNextPoint ::= CHOICE
        {
        distancetonextpointenglish      [0]     DistanceToNextPointEnglish,
        distancetonextpointmetric       [1]     DistanceToNextPointMetric
        }

DistanceToNextPointEnglish ::= INTEGER (1..10000~~1024~~)
        --unit = Nautical Mile, Range (1..1000~~1024~~), resolution = 0.1

DistanceToNextPointMetric ::= INTEGER (1..20000~~2048~~)
        --unit = Kilometer, Range (1..2000~~2048~~), resolution = 0.1

ErrorInformation ::=  ENUMERATED
        {
        unrecognizedMsgReferenceNumber          (0),
        endServiceWithPendingMsgs               (1),
        logicalAcknowledgmentNotAccepted        (2),
        moreThanOneNextDataAuthorityElement     (3),
        ...
        }

FixName ::= IA5String (SIZE (1..5))
```

```
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 (14000~~4680~~..17000~~5520~~)
--      unit = Megahertz, Range (117.00000..138.00000), resolution = 0.00833~~0.025~~

FurtherInstructions ::= SEQUENCE
        {
        beaconCode              [0]     BeaconCode              OPTIONAL,
        frequencyDeparture      [1]     ICAOUnitNameFrequency   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,
        aTWaltitude                     [2]     ATWAltitude     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,
          altitude                    [1]_____Altitude,
          degrees                     [2]_____Degrees,
          direction                   [3]_____Direction,
          legType                     [4]_____LegType           OPTIONAL
          }

ICAOFacilityDesignator ::=  IA5String (SIZE (8))

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

ICAOFacilityDesignatorAltimeter ::= SEQUENCE
          {
          iCAOFacilityDesignator                ICAOUnitName,
          altimeter                             Altimeter
          }

ICAOFacilityDesignatorATISCode ::= SEQUENCE
          {
          iCAOFacilityDesignator                ICAOUnitName,
          aTISCode                              ATISCode
          }

ICAOFacilityIdentification ::=  CHOICE
          {
          iCAOFacilityDesignator                [0]_____ICAOFacilityDesignator,
          iCAOFacilityName                      [1]_____ICAOFacilityName
          }

ICAOFacilityName ::= IA5String (SIZE (3..18))

ICAOUnitName ::=   SEQUENCE
          {
          iCAOFacilityId              ICAOFacilityIdentification,
          iCAOFacilityFunction        ICAOFacilityFunction
          }

ICAOUnitNameFrequency ::=  SEQUENCE
          {
          iCAOUnitName                          ICAOUnitName,
          frequency                   Frequency
          }

InterceptCourseFrom ::= SEQUENCE
          {
          fromSelection           InterceptCourseFromSelection,
          degrees                 Degrees
          }

InterceptCourseFromSelection ::= CHOICE
          {
          publishedIdentifier         [0]     PublishedIdentifier,
          latitudeLongitude           [1]     LatitudeLongitude,
          placeBearingPlaceBearing    [2]     PlaceBearingPlaceBearing,
          placeBearingDistance        [3]     PlaceBearingDistance
          }

Icing ::= ENUMERATED
          {
          trace           (0),
          light           (1),
          moderate        (2),
          severe          (3)
          }


Latitude ::= SEQUENCE
          {
          latitudeDegrees         [0]     LatitudeDegrees,
          minutesLatLon           [1]     MinutesLatLon           OPTIONAL,
          ~~        secondsLatLon           [2]     SecondsLatLon           OPTIONAL,~~
          latitudeDirection       [3]     LatitudeDirection
          }

LatitudeDegrees  ::= INTEGER (0..90000)
          -- unit = Degree, Range (0..90), resolution = 0.001

LatitudeDirection ::= ENUMERATED
          {
          north           (0),
          south           (1)
          }

LatitudeLongitude ::= SEQUENCE
          {
          latitude        Latitude,
          longitude       Longitude
          }

LatitudeReportingPoints ::= SEQUENCE
          {
          latitudeDirection       LatitudeDirection,
          latitudeDegrees         LatitudeDegrees
          }

LatLonReportingPoints ::= CHOICE
          {
          latitudeReportingPoints         [0]     LatitudeReportingPoints,
          longitudeReportingPoints        [1]     LongitudeReportingPoints
          }

LegDistance ::= CHOICE
          {
          legDistanceEnglish              [0]     LegDistanceEnglish,
          legDistanceMetric               [1]     LegDistanceMetric
          }

LegDistanceEnglish ::= INTEGER (0..50~~1..999~~)
          -- unit = Nautical Mile, Range (0..50~~.1..99.9~~), resolution = 1~~0.1~~

LegDistanceMetric ::= INTEGER (1..128)
          -- unit = Kilometer, Range (1..128), resolution = 1

LegTime ::= INTEGER (0..10~~1..99~~)
          --unit = Minute, Range (0..10~~0.1..9.9~~), resolution = 1~~0.1~~

LegType ::= CHOICE
          {
          legDistance     [0]     LegDistance,
          legTime         [1]     LegTime
          }

LevelsOfFlight ::= CHOICE
          {
          altitude                [0]     Altitude,
          procedureName           [1]     ProcedureName,
          altitudeProcedureName   [2]     AltitudeProcedureName
          }

Longitude ::= SEQUENCE
          {
          longitudeDegrees        [0]     LongitudeDegrees,
          minutesLatLon           [1]     MinutesLatLon           OPTIONAL,
          secondsLatLon           [2]     SecondsLatLon           OPTIONAL,
          longitudeDirection      [3]     LongitudeDirection
          }

LongitudeDegrees ::= INTEGER (0..180000)
          --unit = Degree, Range (0..180), resolution = 0.001~~1~~

LongitudeDirection ::= ENUMERATED
          {
          east            (0),
          west            (1)
          }
```

```
LongitudeReportingPoints ::= SEQUENCE
        {
        longitudeDirection          LongitudeDirection,
        longitudeDegrees            LongitudeDegrees
        }

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 ::=  IA5String (SIZE (1..4) )

PersonsOnBoard ::= INTEGER (1..1024)

PlaceBearing ::= SEQUENCE
        {
        fixName                     [0]     FixName,
        latitudeLongitude           [1]     LatitudeLongitude       OPTIONAL,
        degrees                     [2]     Degrees
        }

PlaceBearingDistance ::= SEQUENCE
        {
        fixName                     [0]     FixName,
        latitudeLongitude           [1]     LatitudeLongitude       OPTIONAL,
        degrees                     [2]     Degrees,
        distance                    [3]     Distance
        }

PlaceBearingPlaceBearing ::= SEQUENCE SIZE (2) OF PlaceBearing

PointAltitude ::= SEQUENCE
        {
        altitudeFlightLevel~~Altitude~~         [0]     AltitudeFlightLevel~~Altitude~~,
        aTWAltitudeTolerance        [1]     ATWAltitudeTolerance    OPTIONAL
        }

PointAltitudeBlock ::= SEQUENCE SIZE (2) OF AltitudeFlightLevel~~Altitude~~

PointDetail ::= SEQUENCE
        {
        latitudeLongitude           [0]     LatitudeLongitude,
        fixName                     [1]     FixName                 OPTIONAL,
        truetrackangle              [2]     DegreesTrue             OPTIONAL,
        distancetonextpoint         [3]     DistanceToNextPoint     OPTIONAL,
        pointAltitude               [4]     PointAltitude           OPTIONAL,
        pointAltitudeBlock          [5]     PointAltitudeBlock      OPTIONAL
        }

Position ::=  CHOICE
        {
```

```
        fixName                 [0]     FixName,
        navaid                  [1]     Navaid,
        airport                 [2]     Airport,
        latitudeLongitude       [3]     LatitudeLongitude,
        placeBearingDistance    [4]     PlaceBearingDistance
        }

PositionAltitude ::= SEQUENCE
        {
        position                Position,
        altitude                Altitude
        }

PositionAltitudeAltitude ::= SEQUENCE
        {
        position                Position,
        altitudes               AltitudeAltitude
        }

PositionAltitudeSpeed ::= SEQUENCE
        {
        positionaltitude        PositionAltitude,
        speed                   Speed
        }

PositionDegrees ::=  SEQUENCE
        {
        position                Position,
        degrees                 Degrees
        }

PositionDistanceOffsetDirection ::=  SEQUENCE
        {
        position                Position,
        distanceOffsetDirection DistanceOffsetDirection
        }

PositionICAOUnitNameFrequency ::= SEQUENCE
        {
        position                Position,
        icaounitname            ICAOUnitName,
        frequency               Frequency
        }

PositionPosition ::=  SEQUENCE SIZE (2) OF Position

PositionProcedureName ::= SEQUENCE
        {
        position                Position,
        procedureName           ProcedureName
        }

PositionReport ::= SEQUENCE
```

```
        {
        positioncurrent                 [0]     Position,
        timeatpositioncurrent           [1]     Time,
        altitude                        [2]     Altitude,
        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,
        supplementaryInformation        [18]    FreeText        OPTIONAL,
        reportedWaypointPosition        [19]    Position        OPTIONAL,
        reportedWaypointTime            [20]    Time            OPTIONAL,
        reportedWaypointAltitude        [21]    Altitude        OPTIONAL
        }

PositionRouteClearance ::= SEQUENCE
        {
        position                Position,
        routeClearance          RouteClearance
        }

PositionSpeed ::= SEQUENCE
        {
        position                Position,
        speed                   Speed
        }

PositionSpeedSpeed ::= SEQUENCE
        {
        position                Position,
        speeds                  SpeedSpeed
        }

PositionTime ::= SEQUENCE
        {
        position                Position,
        time                    Time
        }

PositionTimeAltitude ::= SEQUENCE
        {
        positionTime            PositionTime,
        altitude                Altitude
        }

PositionTimeTime ::= SEQUENCE
        {
        position                Position,
        times                   TimeTime
        }

Procedure ::= IA5String (SIZE (1..6))

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 ::= SEQUENCE
        {
        fixName                 [0]     FixName,
        latitudeLongitude       [1]     LatitudeLongitude       OPTIONAL
        }

RemainingFuel ::= Time

RemainingFuelPersons~~Souls~~OnBoard ::=  SEQUENCE
        {
        remainingFuel   RemainingFuel,
        persons~~souls~~OnBoard       Persons~~Souls~~OnBoard
        }

ReportingPoints ::= SEQUENCE
        {
        latLonReportingPoints   [0]     LatLonReportingPoints,
        degreeIncrement         [1]     DegreeIncrement                 OPTIONAL
        }

RevisionNumber ::= INTEGER (1..16)

RouteAndLevels ::= SEQUENCE
        {
        routeOfFlight           RouteInformation,
        levelsOfFlight          LevelsOfFlight
        }
```

```
RouteClearance ::= SEQUENCE
        {
        airportDeparture                [0]     Airport                         OPTIONAL,
        airportDestination              [1]     Airport                         OPTIONAL,
        runwayDeparture                 [2]     Runway                          OPTIONAL,
        procedureDeparture              [3]     ProcedureName                   OPTIONAL,
        runwayArrival                   [4]     Runway                          OPTIONAL,
        procedureApproach               [5]     ProcedureName                   OPTIONAL,
        procedureArrival                [6]     ProcedureName                   OPTIONAL,
        airwayIntercept                 [7]     AirwayIntercept                 OPTIONAL,
        routeInformations               [8]     SEQUENCE SIZE (1..128)
                                                OF RouteInformation             OPTIONAL,
        routeInformationAdditional      [9]     RouteInformationAdditional      OPTIONAL
        }

RouteInformation ::= CHOICE
        {
        publishedIdentifier             [0]     PublishedIdentifier,
        latitudeLongitude               [1]     LatitudeLongitude,
        placeBearingPlaceBearing        [2]     PlaceBearingPlaceBearing,
        placeBearingDistance            [3]     PlaceBearingDistance,
        airwayIdentifier                [4]     AirwayIdentifier,
        trackDetail                     [5]     TrackDetail
        }

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,
        waypointSpeedAltitudes          [4]     SEQUENCE SIZE (1..32) OF WaypointSpeedAltitude
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

~~RVR ::= CHOICE~~
~~        {~~
~~        rVRFeet         RVRFeet,~~
~~        rVRMeters       RVRMeters~~
~~        }~~

~~RVRFeet ::= INTEGER (0..6100)~~
~~        -- unit = Feet, Range (0..6100), resolution = 1~~

~~RVRMeters ::= INTEGER (0..5000)~~
~~        -- unit = Meters (0..5000), resolution = 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..5000)
        -- unit = Meters (0..5000), 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~~,~~
        ~~speedMachLarge                  SpeedMachLarge~~
        }

SpeedIndicated ::= INTEGER (0..400~~7..38~~)
        -- unit = Knots, Range (0..400~~70..380~~), resolution = 1~~10~~

SpeedIndicatedMetric ::= INTEGER (0..800~~10..137~~)
        -- unit = Kilometers/Hour, Range (0..800~~100..1370~~), resolution = 1~~10~~

SpeedGround ::= INTEGER (-50..2000~~7..70~~)
        -- unit = Knots, Range (-50..2000~~70..700~~), resolution = 1~~10~~

SpeedGroundMetric ::= INTEGER (-100..4000~~10..265~~)
        -- unit = Kilometers/Hour, Range (-100..4000~~100..2650~~), resolution = 1~~10~~

SpeedMach ::= INTEGER (50..400~~61..92~~)
        -- unit = Mach Range (0.5 to 4.0~~.61 to .92~~), resolution = 0.001

~~SpeedMachLarge ::= INTEGER (93..604)~~
~~        -- unit = Mach Range (.93 to 6.04), resolution = 0.01~~

SpeedQualifier ::= SEQUENCE
        {
        speedQualifiers   SpeedQualifiers   OPTIONAL
        }

SpeedQualifiers ::= ENUMERATED
        {
        approach        (0),
        minimum         (1),
        cruise          (2),
        ...
        }

SpeedQualifierSpeedType ::= SEQUENCE
        {
        speedQualfier           SpeedQualifier, jane make optiona and simplify
        speedType               SpeedType
        }

SpeedSpeed ::=  SEQUENCE SIZE (2) OF Speed
```

SpeedTime ::= SEQUENCE
{
speed		Speed,
time		Time
}

SpeedTrue ::= INTEGER (7..70)
-- unit = Knots, Range (70..700), resolution = 10

SpeedTrueMetric ::= INTEGER (10..137)
-- unit = Kilometers/Hour, Range (100..1370), resolution = 10

SpeedType ::= SEQUENCE
{
speedTypes	SpeedTypes	OPTIONAL
}

SpeedTypes ::= ENUMERATED
{
indicated	(0),
true	(1),
ground	(2),
mach	(3),
machLarge	(4)
}

~~**SoulsOnBoard** ::= INTEGER (1..1024)~~

SSREquipmentAvailable ::= ENUMERATED
{
nnil	(0),
atransponderModeA	(1),
ctransponderModeAandC	(2),
xtransponderModeS	(3),
ptransponderModeSPA	(4),
itransponderModeSID	(5),
stransponderModeSPAID	(6)~~,~~
...
}
--*Note: PA Pressure Altitude; ID Aircraft Identification*

~~**Temperature** ::= CHOICE~~
~~{~~
~~temperatureC	TemperatureC,~~
~~temperatureF	TemperatureF~~
~~}~~

Temperature~~C~~ ::= INTEGER (-100..100~~-80..47~~)
-- unit = Degree Centigrade, Range (-100..100~~-80..47~~), resolution = 1

~~**TemperatureF** ::= INTEGER (-105..150)~~
~~-- unit = Degree Fahrenheit, Range (-105..150), resolution = 1~~

```
Time ::= SEQUENCE
        {
        hours                   TimeHours,
        minutes                 TimeMinutes
        }

TimeAltitude ::= SEQUENCE
        {
        time            Time,
        altitude        Altitude
        }

TimeDeparture ::= SEQUENCE
        {
        timeDepartureAllocated          [0]     Time                          OPTIONAL,
        timeDepartureControlled         [1]     ControlledTime                OPTIONAL,
        timeDepartureClearanceExpected  [2]     Time                          OPTIONAL,
        departureMinimumInterval        [3]     DepartureMinimumInterval      OPTIONAL
        }

TimeDistanceOffsetDirection ::= SEQUENCE
        {
        time                    Time,
        distanceOffsetDirection DistanceOffsetDirection
        }

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

TimeICAOUnitNameFrequency ::=  SEQUENCE
        {
        time                    Time,
        iCAOUnitName            ICAOUnitName,
        frequency               Frequency
        }

TimeMinutes ::= INTEGER (0..59)
        -- unit = Minute, Range (0..59), resolution = 1
```

```
TimePosition ::=  SEQUENCE
        {
        time                    Time,
        position                Position
        }

TimePositionAltitude ::=    SEQUENCE
        {
        timeposition            TimePosition,
        altitude                Altitude
        }

TimePositionAltitudeSpeed ::= SEQUENCE
        {
        timeposition            TimePosition,
        altitudespeed           AltitudeSpeed
        }

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
        }

TrackDetail ::= SEQUENCE
        {
        trackName               TrackName,
        latitudeLongitudes      SEQUENCE SIZE (1..128) OF LatitudeLongitude
        }

TrackDetailMsg ::= SEQUENCE
        {
        trackname               TrackName,
        datetimetrackgenerated  DateTimeGroup,
        trackDetailMsgType      TrackDetailMsgType,
        datetimetrackstart      DateTimeGroup,
        datetimetrackstop       DateTimeGroup,
        airportdeparture        Airport,
        pointdetails            SEQUENCE SIZE (1..32) OF PointDetail,
        airportdestination      Airport,
        remarks                 FreeText
        }

TrackDetailMsgType ::= ENUMERATED
        {
        provisional     (0),
        final           (1)
        }

TrackName ::= IA5String (SIZE (3..6))

TrafficType ::= SEQUENCE
        {
        trafficTypes    TrafficTypes    OPTIONAL
        }

TrafficTypes ::= ENUMERATED
        {
        oppositeDirections      (0),
        sameDirectional         (1),
        converging              (2),
        crossing                (3)
        }

Turbulence ::= ENUMERATED
        {
        light           (0),
        moderate        (1),
```

    severe   (2)
    }

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..300~~60~~)
    -- unit = Feet/Minute, Range (0..30000~~6000~~), resolution = 10~~100~~

VerticalRateMetric ::= INTEGER (0..1000~~200~~)
    -- unit = Meters/Minute, Range (0..10000~~2000~~), resolution = 10

WaypointSpeedAltitude ::= SEQUENCE
    {
    position   [0] Position,
    speed   [1] Speed   OPTIONAL,
    aTWAltitudes   [2] ATWAltitudeSequence   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 (0..99)
        -- unit = Year, Range (0..99), resolution = 1


END
```

5. PROTOCOL DEFINITION

5.1 Sequence Rules

Note: — 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.

5.1.1 With the exception of abort primitives, only the sequence of primitives described below shall be permitted.

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

5.1.2 With the exception of abort primitives, the CPDLC-air-ASE and CPDLC-ground-ASE shall process primitives in the order in which they are received.

Note: — This ensures that the CPDLC-ASE will guarantee message sequencing, with the exception of aborts.

5.1.3 CPDLC-start Service

5.1.3.1 Air Initiated

Note: — The following sequence of messages, shown in Figure 5-1, occurs when the CPDLC-Start service is air initiated.

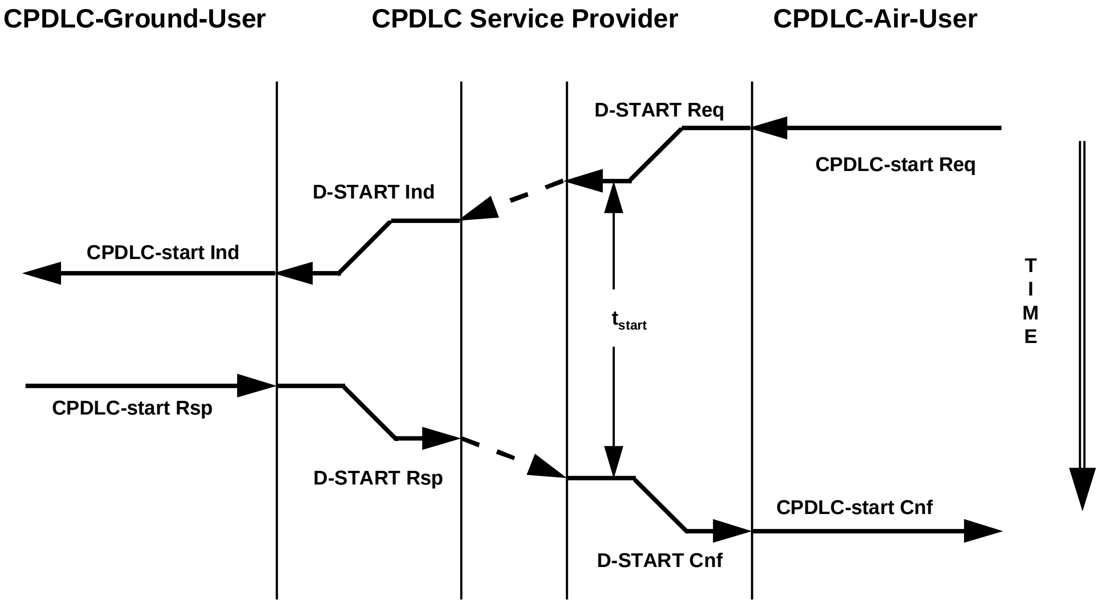

Figure 5-1: Sequence Diagram for CPDLC-start Service Air Initiated

5.1.3.2 Ground Initiated

Note: — The following sequence of messages, shown in Figure 5-2, occurs when the CPDLC-start service is ground initiated.

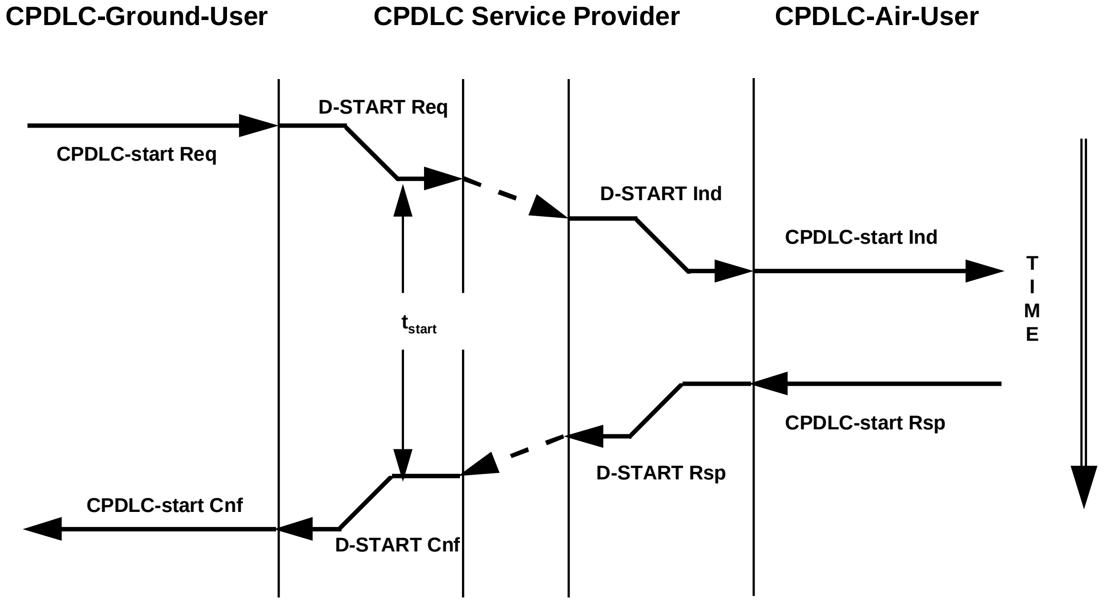

Figure 5-2: Sequence Diagram for CPDLC-start Service Ground Initiated

5.1.4 DSC-start Service

Note: —The following sequence of messages, shown in Figure 5-3, occurs when the DSC-start service is initiated.

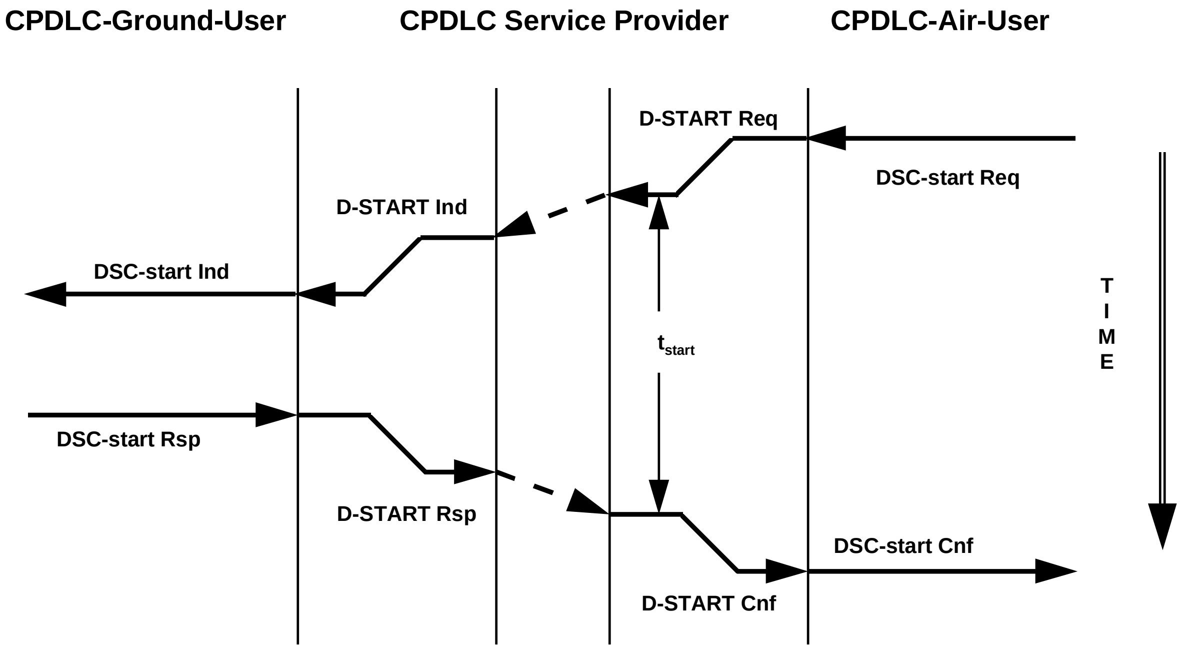

Figure 5-3: Sequence Diagram for DSC-start Service

5.1.5 CPDLC-message Service

5.1.5.1 Air Initiated

Note:— The following sequence of messages, shown in Figure 5-4, occurs when the CPDLC-message Service is air initiated.

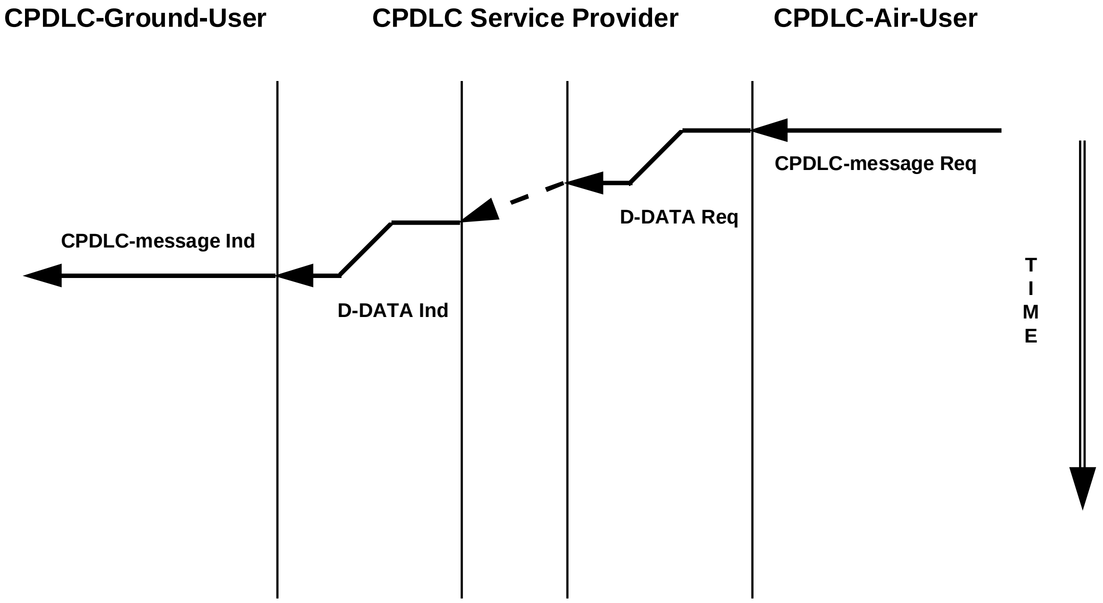

Figure 5-4: Sequence Diagram for CPDLC-message Service Air Initiated

5.1.5.2 Ground Initiated

Note: — The following sequence of messages, shown in Figure 5-5, occurs when the CPDLC-message Service is ground initiated.

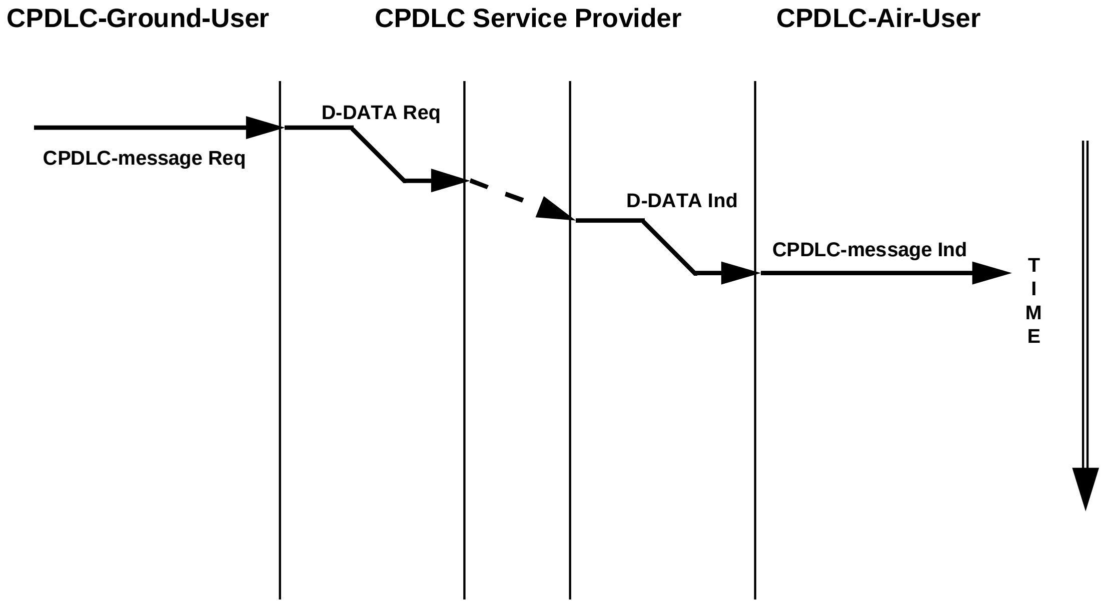

Figure 5-5: Sequence Diagram for CPDLC-message Service Ground Initiated

5.1.6 CPDLC-end Service

Note:. — The following sequence of messages, shown in Figure 5-6, occurs when the CPDLC-end service is initiated.

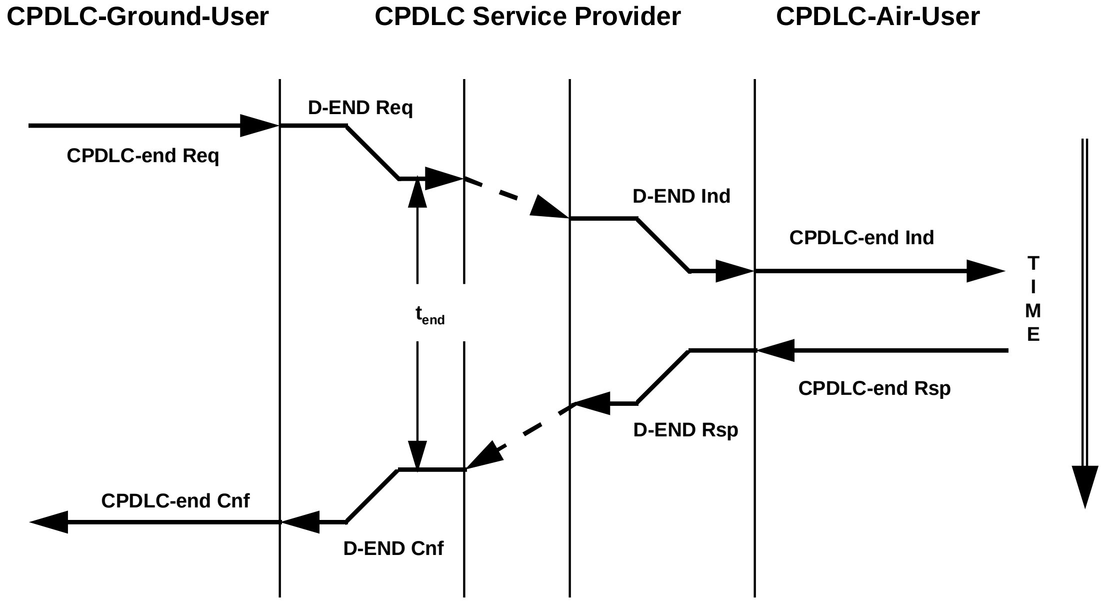

Figure 5-6: Sequence Diagram for CPDLC-end Service

5.1.7 DSC-end Service

Note: — The following sequence of messages, shown in Figure 5-7, occurs when the DSC-end service is initiated.

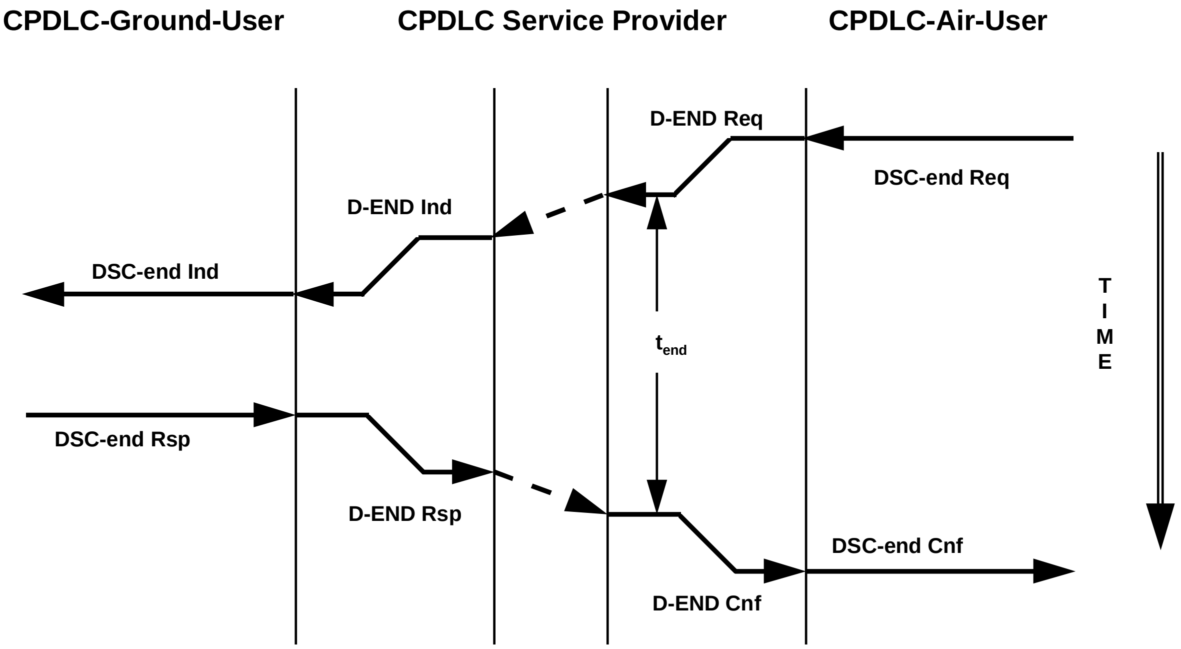

Figure 5-7: Sequence Diagram for DSC-end Service

5.1.8 CPDLC-forward Service

Note: — The following sequence of messages, shown in Figure 5-8, occurs when the CPDLC forward service is initiated.

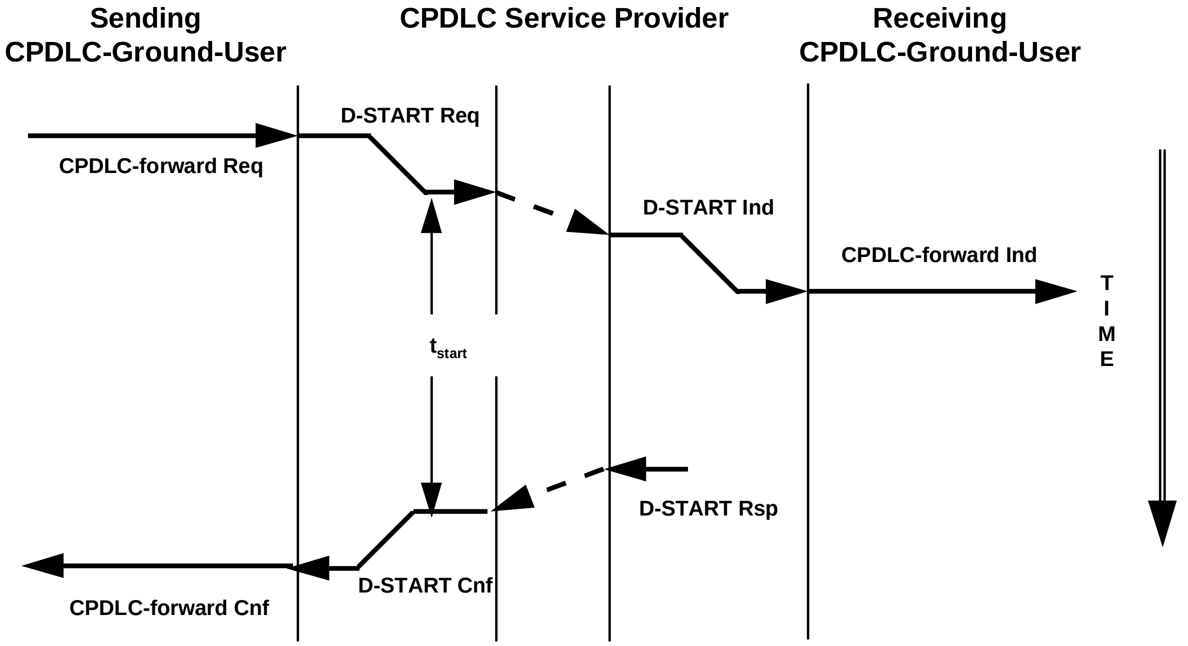

Figure 5-8: Sequence Diagram for CPDLC-forward Service

5.1.9 CPDLC-user-abort-Service

5.1.9.1 CPDLC-Air-User Initiated

Note: — The following sequence of messages, shown in Figure 5-8, occurs when the CPDLC-user-abort service is initiated by the CPDLC-air-user.

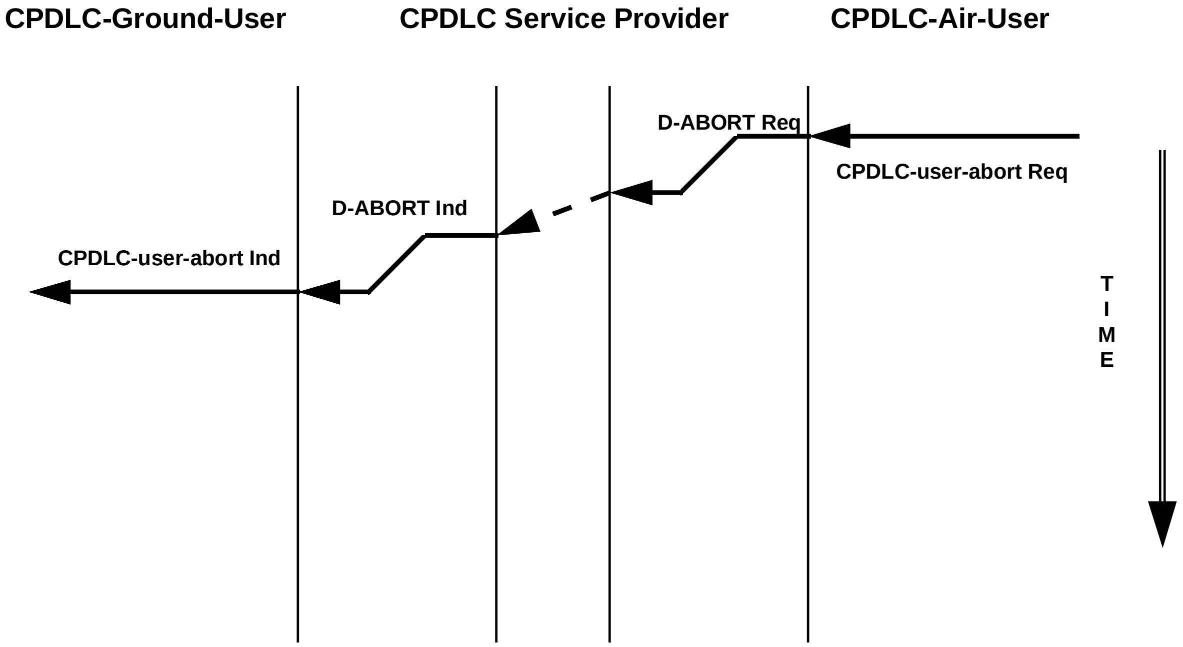

Figure 5-8: Sequence Diagram for CPDLC-user-abort Service CPDLC-Air-User Initiated

5.1.9.2 CPDLC-Ground-User Initiated

Note: — The following sequence of messages, shown in Figure 5-9, occurs when the CPDLC-user-abort service is initiated by the CPDLC-ground-user.

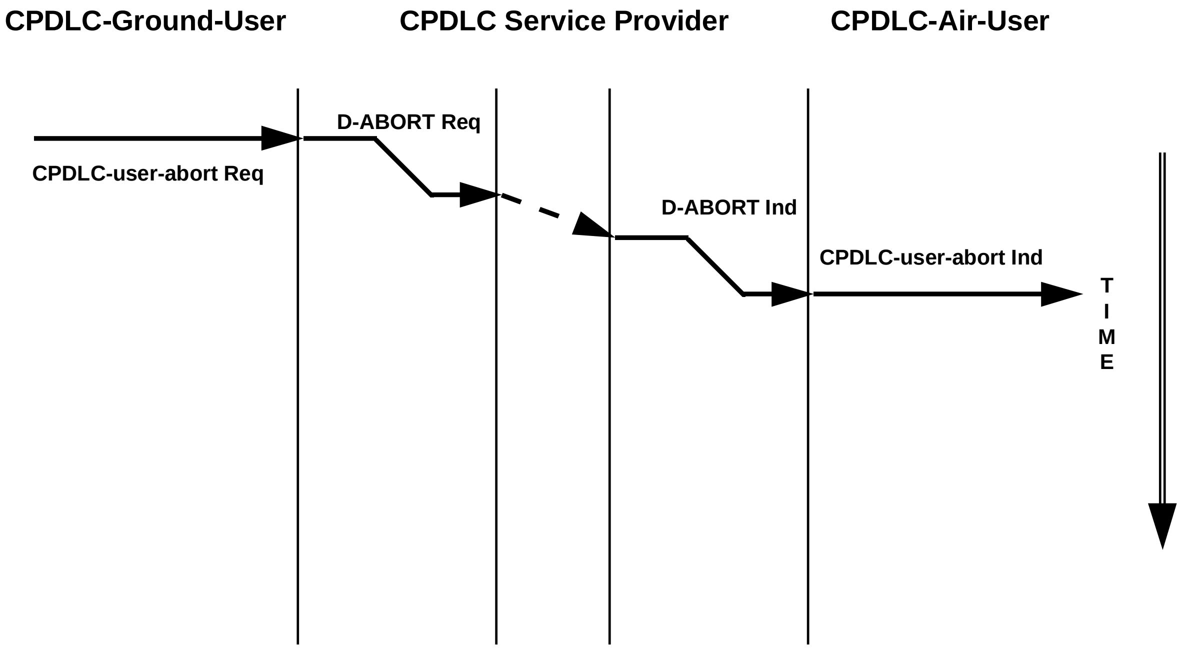

Figure 5-9: Sequence Diagram for CPDLC-user-abort Service CPDLC-Ground-User Initiated

5.1.10 CPDLC-provider-abort-Service

5.1.10.1 Dialogue Service Abort

Note: — The following sequence of messages, shown in Figure 5-10, occurs when the dialogue service provider (below the level of the ASE) aborts.

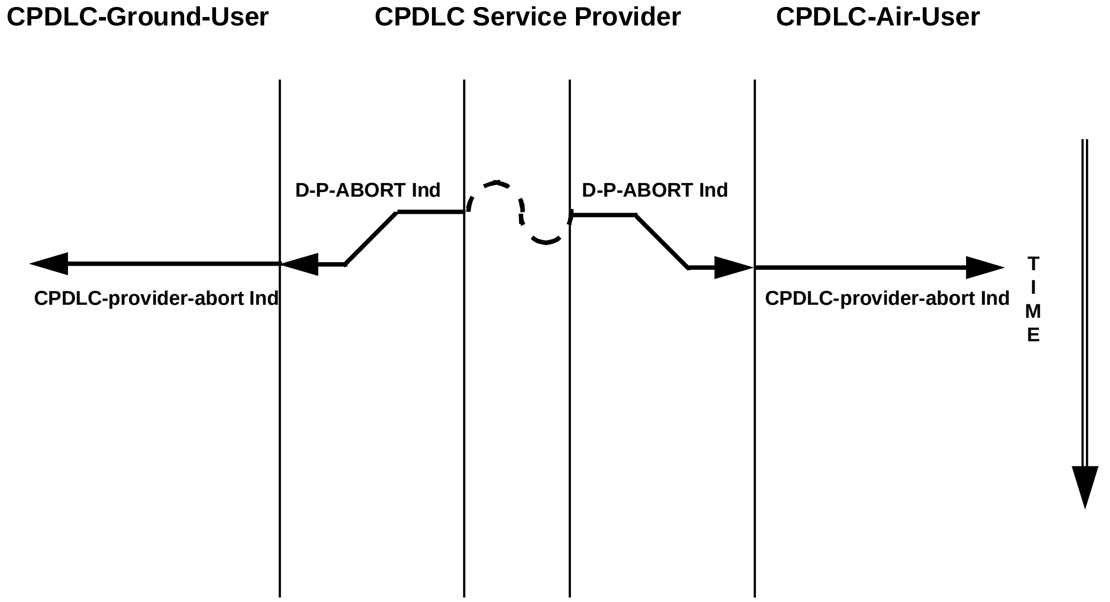

Figure 5-10: Sequence Diagram for CPDLC-provider-abort Service Dialogue Service Abort

5.1.10.2 CPDLC-Air-ASE Abort

Note: — The following sequence of messages, shown in Figure 5-11, occurs when the CPDLC-air-ASE aborts.

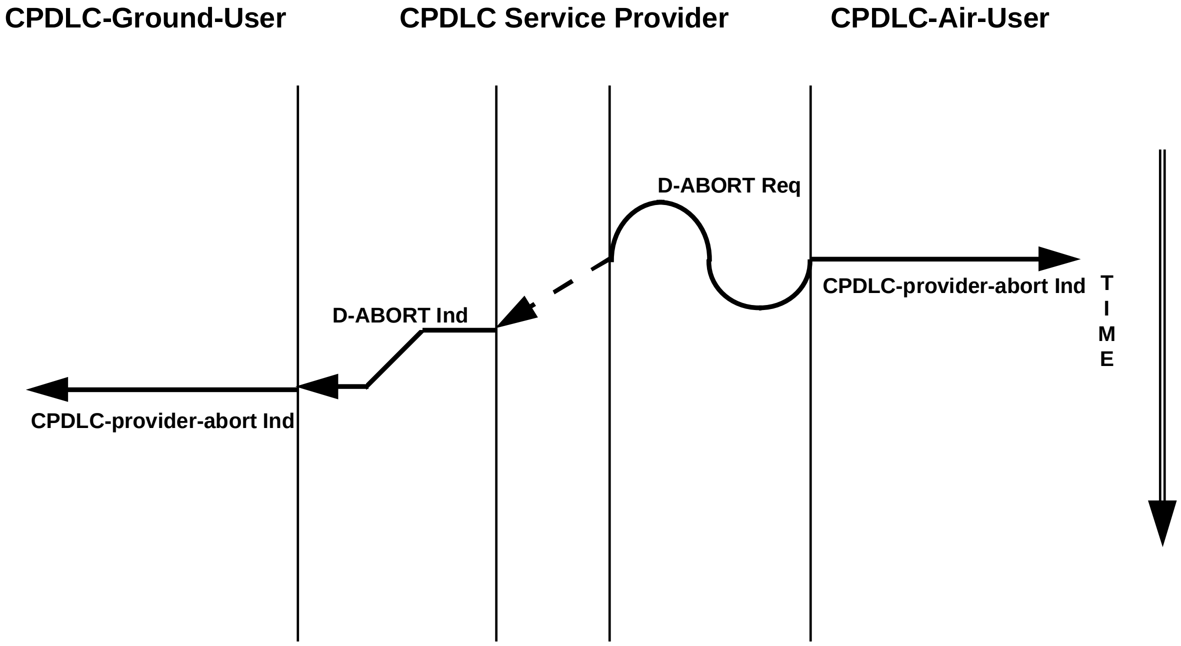

Figure 5-11: Sequence Diagram for CPDLC-provider-abort Service CPDLC-Air-ASE Abort

5.1.10.3 CPDLC-Ground-ASE Abort

Note: — The following sequence of messages, shown in Figure 5-12, occurs when the CPDLC-ground-ASE aborts.

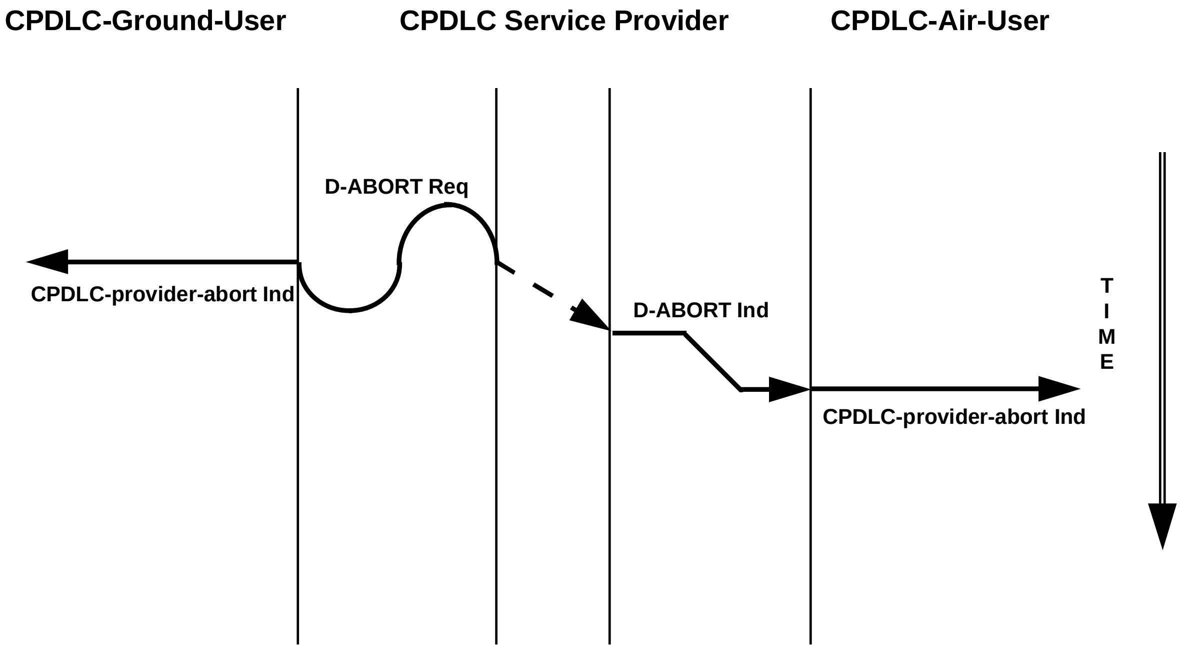

Figure 5-12: Sequence Diagram for CPDLC-provider-abort Service CPDLC-Ground-ASE Abort

5.1.11 CPDLC-user-abort-Service (Ground-Ground)

5.1.11.1 Receiving CPDLC-Ground-User Initiated

Note: — The following sequence of messages, shown in Figure 5-13, occurs when the CPDLC-user-abort service is initiated by the receiving CPDLC-ground-user.

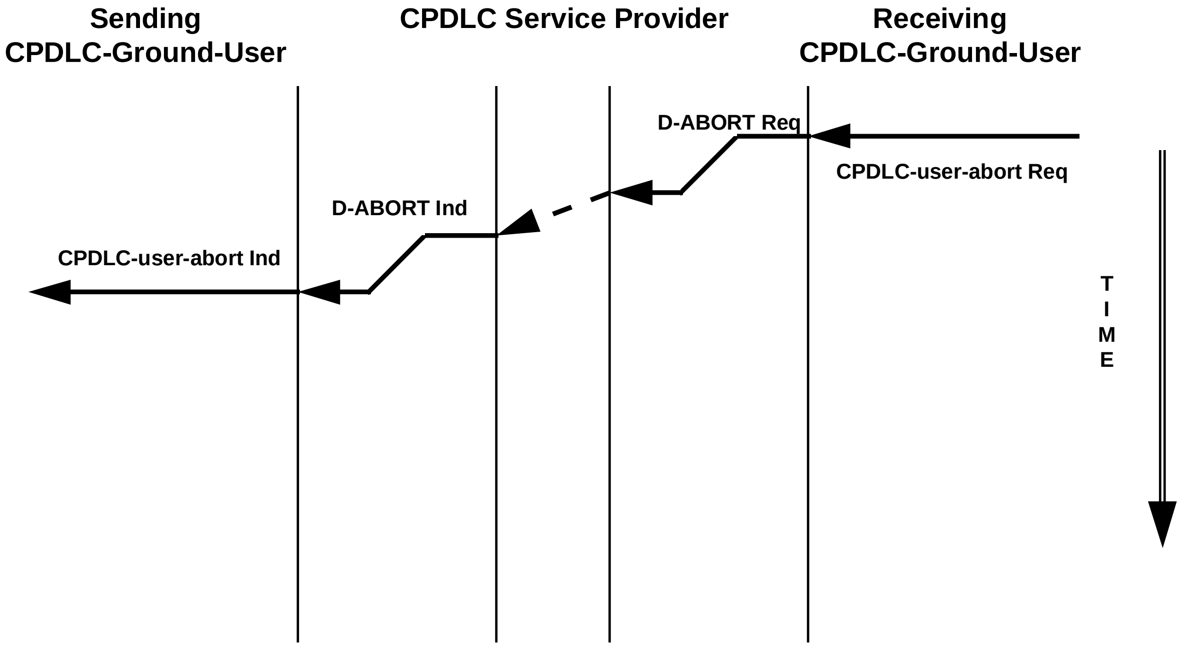

*Figure 5-13: Sequence Diagram for CPDLC-user-abort Service
Receiving CPDLC-Ground-User Initiated*

5.1.11.2 Sending CPDLC-Ground-User Initiated

Note: — The following sequence of messages, shown in Figure 5-14, occurs when the CPDLC-user-abort service is initiated by the sending CPDLC-ground-user.

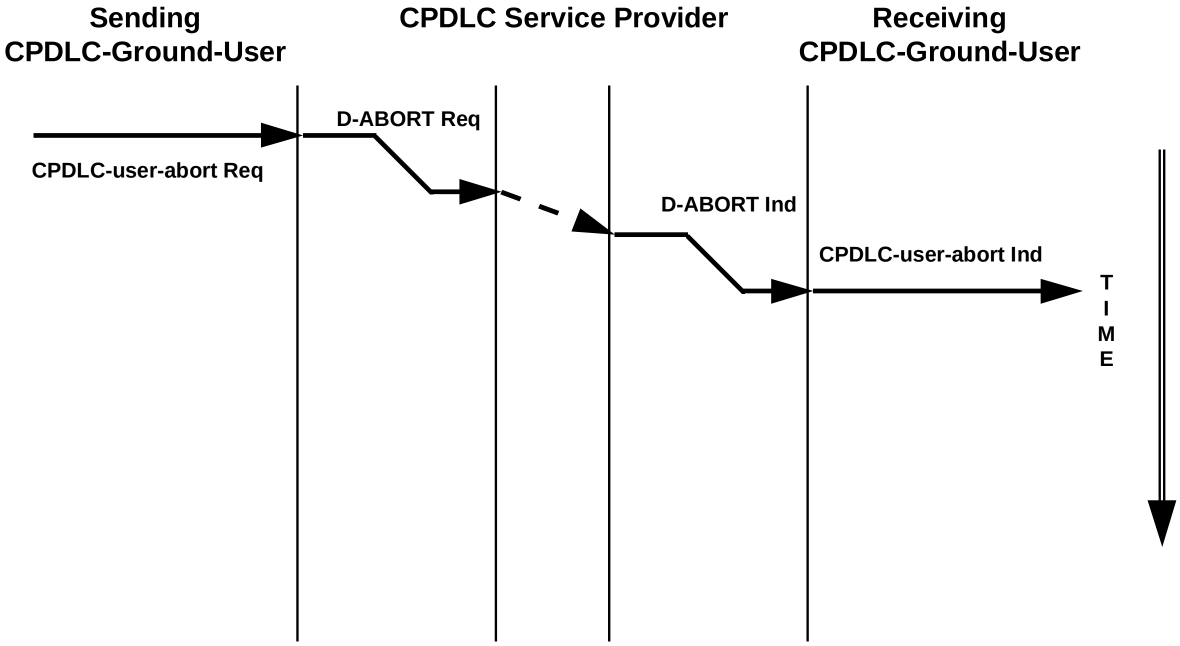

Figure 5-14: Sequence Diagram for CPDLC-user-abort Service Sending CPDLC-Ground-User Initiated

5.1.12 CPDLC-provider-abort-Service (Ground-Ground)

5.1.12.1 Dialogue Service Abort

Note: — The following sequence of messages, shown in Figure 5-15, occurs when the dialogue service provider (below the level of the ASE) aborts.

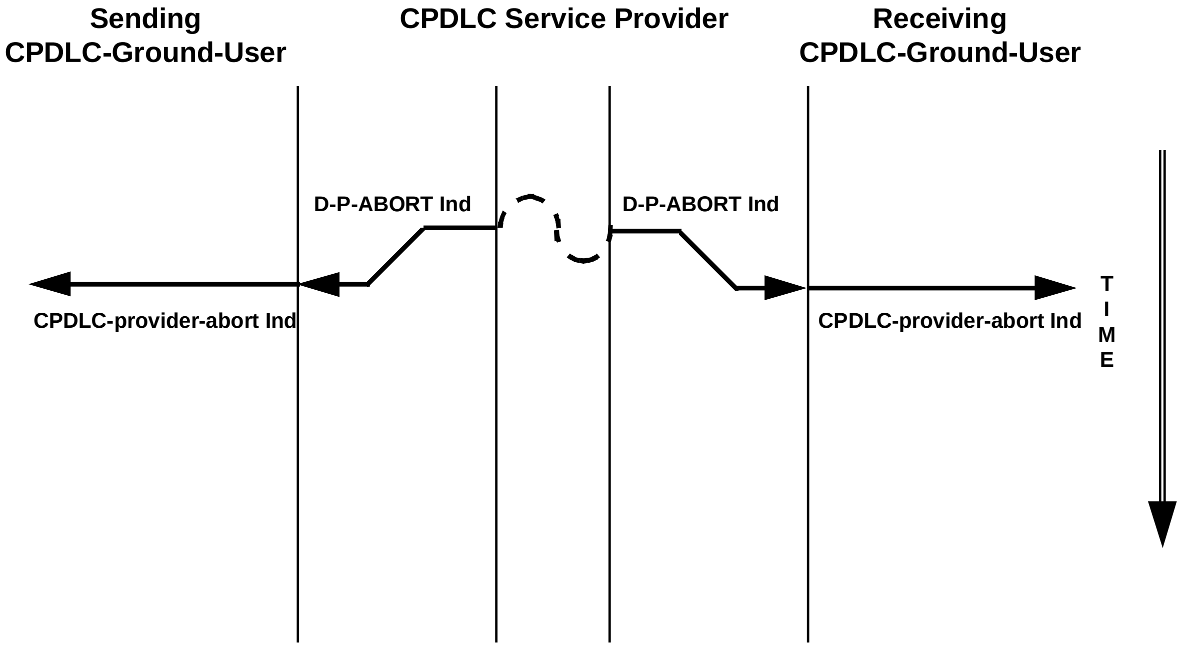

Figure 5-15: Sequence Diagram for CPDLC-provider-abort Service Dialogue Service Abort

5.1.12.2 Receiving CPDLC-Ground-ASE Abort (Ground-Ground)

Note: — The following sequence of messages, shown in Figure 5-16, occurs when the receiving CPDLC-ground-ASE aborts.

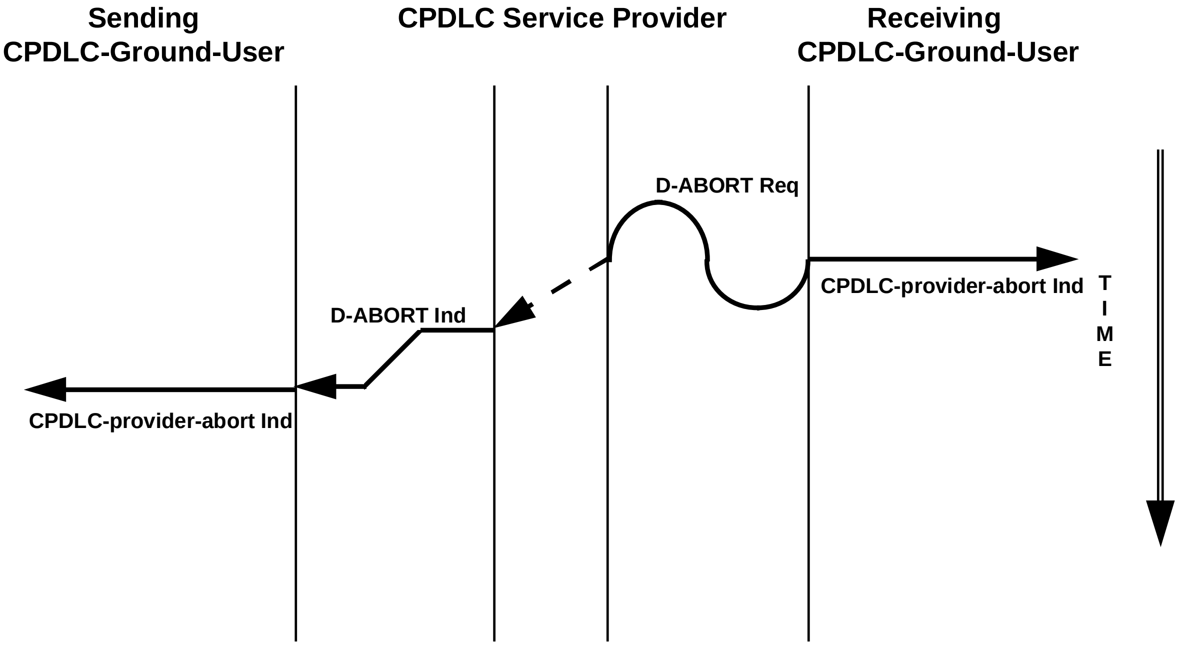

Figure 5-16: Sequence Diagram for CPDLC-provider-abort Service Receiving CPDLC-Ground-ASE Abort

5.1.12.3 Sending CPDLC-Ground-ASE Abort (Ground-Ground)

Note: — The following sequence of messages, shown in Figure 5-17, occurs when the sending CPDLC-ground-ASE aborts.

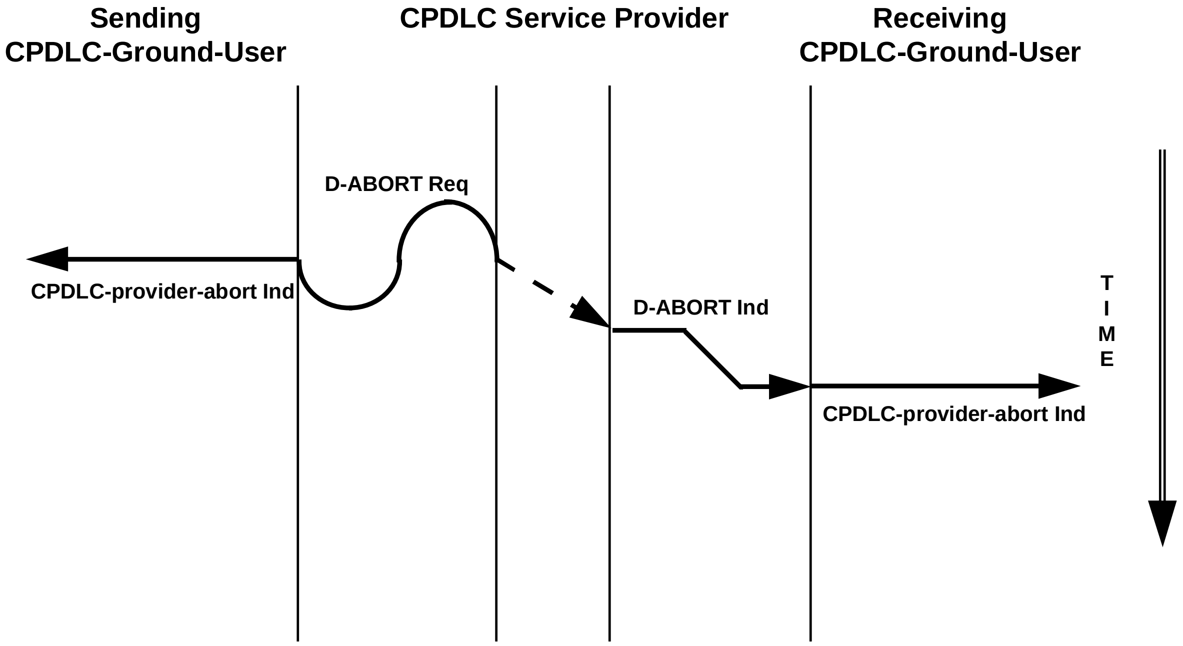

Figure 5-17: Sequence Diagram for CPDLC-provider-abort Service Sending CPDLC-Ground-ASE Abort

5.2 CPDLC Service Provider Timers

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

Note 1: — Table 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.

~~A CPDLC-ASE shall measure the time between the initial event and the corresponding final event.~~

Note 2: — If the timer expires ~~maximum time is exceeded~~ before the final event has occurred, a CPDLC-ASE shall take appropriate action as defined in section 5.4.1~~3~~.

5.2.2 The ~~action shall be taken when the maximum~~ timer values should be as indicated in Table 5-1~~ has expired~~.

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

Table 5-1: CPDLC Service Provider~~Application~~ Timers

5.3 CPDLC ASE Protocol Description

5.3.1 Introduction

Note: — This section presents requirements for CPDLC-ASEs in specific states. Section 5.5 contains state tables for the CPDLC ASEs.

5.3.1.1 Where requirements are stated for a CPDLC-ASE in this section, they shall apply to both the CPDLC-air-ASE and the CPDLC-ground-ASE.

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

5.3.1.3 Upon receipt of a PDU, i~~I~~f no actions are described for the arrival of that ~~a~~ PDU when a CPDLC-ASE is in a specific state, then exception handling procedures as described in section 5.4.4 shall apply.

Note: — The states defined for the CPDLC-ASE are the following:

a) IDLE
b) START-REQ,
c) START-IND,
d) DIALOGUE,
e) END, and
f) FOWARD.

5.3.1.4 On initiation the CPDLC-ASE shall be in the IDLE state.

Note: — The CPDLC-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.

5.3.1.5 On the initiation of a CPDLC-ASE, DSC shall be set to the abstract value “false”.

5.3.2 **D-START Indication**

5.3.2.1 Upon receipt of a D-START indication, if the CPDLC-ASE is in the *IDLE* state, and the D-START *User Data* parameter contains a GroundPDUs APDU and the APDU element is not an ATCForwardMessage or contains an AircraftPDUs APDU with the APDU-element mode “cpdlc”, the CPDLC-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) if the AircraftPDUs or GroundPDUs APDU-element contained in the D-START *User Data* parameter is either an ATCUplinkMessage or an ATCDownlinkMessage, set the AircraftPDUs or GroundPDUs APDU-element as the CPDLC-start service *CPDLC Message* parameter value, and

b) Enter the *START-IND* state.

5.3.2.2 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the D-START *User Data* parameter contains a GroundPDUs APDU and the APDU element is an ATCForwardMessage the CPDLC-ground-ASE shall:

a) Invoke CPDLC-forward service indication containing the following:
   1) the D-START *Calling Peer ID* parameter value as the CPDLC-forward service *Calling ICAO Facility Designator* parameter value,
   2) set the GroundPDUs APDU-element as the CPDLC-forward service *CPDLC Message* parameter value, and

b) Invoke D-START response with the abstract value “rejected (permanent)” as the D-START *Result* parameter value, and

c) Enter the *IDLE* state.

5.3.2.3 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the D-START *User Data* parameter contains an AircraftPDUs APDU with the APDU-element mode “dsc”, 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 Identifier* parameter value,
   2) if the APDU AircraftPDUs APDU contained in the D-START is an ATCDownlinkMessage, set the AircraftPDUs APDU as the DSC-start service *CPDLC Message* parameter value,

b) Set DSC to “true”, and

c) Enter the *START-IND* state.

5.3.3 **D-START Confirmation**

5.3.3.1 Upon receipt of a D-START confirmation, if the CPDLC-ASE is in the *START-REQ* state and if either of the D-START *Result* and *Reject Source* parameters do not have the abstract values “rejected (transient)” and “DS provider” respectively , and DSC has the abstract value of “false”, the CPDLC-ASE shall:

a) Stop timer t_{start},

b) Invoke CPDLC-start service confirmation containing the following:
   1) the AircraftPDUs or GroundPDUs APDU-element contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, if provided, and
   2) the D-START *Result* parameter value as the CPDLC-start service *Result* parameter value as follows:
      i) “accepted” as “accepted”, or
      ii) “rejected (permanent)” as “rejected”, and

c) If the D-START *Result* parameter abstract value is not “accepted” enter the *IDLE* state, or

d) Else enter the *DIALOGUE* state.

5.3.3.2 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and if either of the D-START *Result* and *Reject Source* parameters do not have the abstract values “rejected (transient)” and “DS provider” respectively , and DSC has the abstract value of “true”, the CPDLC-air-ASE shall:

a) Stop timer t_{start},
b) Invoke DSC-start service confirmation containing the following:
  1) the GroundPDUs APDU-element contained in the D-START *User Data* parameter as the DSC-start service *Reject Reason* parameter value, if provided, and
  2) the D-START *Result* parameter value as the DSC-start *Result* parameter value as follows:
    i) “accepted” as “accepted”, or
    ii) “rejected (permanent)” as “rejected”, and
c) If the D-START *Result* parameter abstract value is not “accepted” enter the *IDLE* state, or
d) Else enter the *DIALOGUE* state.

5.3.3.3 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”, the CPDLC-ground-ASE shall:

a) Stop timer t_{start},
b) Invoke CPDLC-forward service confirmation, and
c) Enter the *IDLE* state.

5.3.4 D-DATA Indication

5.3.4.1 Upon receipt of a D-DATA indication, if the CPDLC-ASE is in the *DIALOGUE* state, the CPDLC-ASE shall:

a) Invoke CPDLC-message service indication with the APDU contained in the D-DATA *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, and
b) Remain in the *DIALOGUE* state.

5.3.4.2 Upon receipt of a D-DATA indication, if the CPDLC-ground-ASE is in the END state, the CPDLC-ground-ASE shall:

a) Invoke CPDLC-message service indication with the APDU contained in the D-DATA *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, and
b) Remain in the *END* state.

5.3.5 D-END Indication

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

a) Invoke CPDLC-end service indication with the APDU contained in the D-END *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, if provided, as the CPDLC-end *CPDLC Message* parameter value, and
b) Enter the *END* state

5.3.5.2 Upon receipt of a D-END indication, if the CPDLC-ground-ASE is in the *DIALOGUE* state, the CPDLC-ground-ASE shall:

a) Invoke DSC-end service indication with the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, if provided, and
b) Enter the *END* state

5.3.6 D-END Confirmation

5.3.6.1 Upon receipt of a D-END confirmation, if the CPDLC-ground-ASE is in the *END* state and the abstract values of either of the D-END *Result* and *Reject Source* are not “rejected (transient)” or “DS provider” respectively, the CPDLC-ground-ASE shall:

a) Stop timer t_{end},

b) Invoke CPDLC-end service confirmation with:
    1) The APDU contained in the D-END *User Data* parameter as the CPDLC-end service *CPDLC Message* parameter value, if provided, and
    2) The D-END *Result* parameter value as the CPDLC-end service *Result* parameter value as follows:
        i) "accepted as "accepted", or
        ii) "rejected (permanent)" as "rejected", and

c) If the D-END *Result* is "rejected (permanent)" enter the *DIALOGUE* state, or

d) Else enter the *IDLE* state.

5.3.6.2 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the abstract values of either of the D-END *Result* and *Reject Source* are not "rejected (transient)" or "DS provider" respectively, the CPDLC-air-ASE shall:

a) Stop timer t_{end},

b) Invoke DSC-end service confirmation with:
    1) The APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, if provided, and
    2) The D-END *Result* parameter value as the DSC-end service *Result* parameter value as follows:
        i) "accepted as "accepted", or
        ii) "rejected (permanent)" as "rejected", and

c) If the D-END *Result* is "rejected (permanent)" enter the *DIALOGUE* state, or

d) Else enter the *IDLE* state.

5.3.7 **CPDLC-start Service Request**

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

a) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with a StartDownMessage APDU element containing:
    1) the abstract value "cpdlc" as the mode,
    2) if provided, the *CPDLC Message* parameter as the DownlinkMessage, or
    3) else, NULL as the DownlinkMessage,

b) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with an UplinkMessage APDU element containing:
    1) if provided, the *CPDLC Message* parameter as the UplinkMessage, or
    2) else, NULL as the UplinkMessage,

c) Invoke D-START request with the following:
    1) the CPDLC-start service *Called Peer Identifier* parameter value as the D-START *Called Peer ID* parameter value,
    2) the CPDLC-start service *Calling Peer Identifier* parameter value as the D-START *Calling Peer ID* parameter value,
    3) the D-START *Quality of Service* parameters set as follows:
        i) if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START QOS *Routing Class* parameter value, or
        ii) The abstract value of "normal priority flight safety messages", as the D-START *QOS Priority* parameter value, and
        iii) The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and
    4) if the CPDLC-ASE is a CPDLC-air-ASE, the AircraftPDUs APDU as the D-START *User Data* parameter value;
    5) if the CPDLC-ASE is a CPDLC-ground-ASE, the GroundPDUs APDU as the D-START *User Data* parameter value;

d) Start timer t_{start}, and

e) Enter the *START-REQ* state.

5.3.8 CPDLC-start Service Response

5.3.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-ASE is in the *START-IND* state, the CPDLC-ASE shall:

a) If the CPDLC-ASE is a CPDLC-air-ASE, and the CPDLC-start service *Reject Reason* parameter is provided, create an AircraftPDUs APDU with an ATCDownlinkMessage APDU element based on the *Reject Reason* parameter,
b) If the CPDLC-ASE is a CPDLC-ground-ASE, and the CPDLC-start service *Reject Reason* parameter is provided, create a GroundPDUs APDU with an ATCUplinkMessage APDU element based on the *Reject Reason* parameter,
c) Invoke D-START response with the following:
    1) The APDU as the D-START *User Data* parameter value; if the *Reject Reason* parameter was provided by the CPDLC-user, and
    2) the CPDLC-start service response *Result* parameter value as the D-START *Result* parameter value, and
d) If the abstract value of the *Result* parameter is not “accepted” enter the *IDLE* state, or
e) Else enter the *DIALOGUE* state.

5.3.9 DSC-start Service Request

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 ATCDownlinkMessage APDU element containing:
    1) the abstract value “dsc” as the mode,
    2) if provided, the *CPDLC Message* parameter as the DownlinkMessage, or
    3) else, NULL as the DownlinkMessage,
b) Invoke D-START request with the following:
    1) the DSC-start service *ICAO Facility Designator* parameter value as the D-START *Called Peer ID* parameter value,
    2) the DSC-start service *Aircraft Identifier* parameter value as the D-START *Calling Peer ID* parameter value,
    3) Set the D-START *Quality of Service* parameters as follows:
        i) *Class of Communication* parameter from the DSC-start service request if provided by the CPDLC-air-user,
        ii) The abstract value of “normal priority flight safety messages”, as the D-START *QOS Priority* parameter value, and
        iii) The abstract value of “low” as the D-START *QOS Residual Error Rate* parameter value, and
    4) the APDU as the D-START *User Data* parameter value;
c) Set DSC to “true”,
d) Start timer t_{start}, and
e) Enter the *START-REQ* state.

5.3.10 DSC-start Service Response

5.3.10.1 Upon receipt of a DSC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state, the CPDLC-ground-ASE shall:

a) If the DSC-start service *Reject Reason* parameter is provided, create a GroundPDUs APDU with an ATCUplinkMessage APDU element based on the *Reject Reason* parameter,
b) Invoke D-START response with the following:
    1) The APDU element as D-START *User Data* parameter value; if the *Reject Reason* parameter was provided by the CPDLC-ground-user, and
    2) The DSC-start service response *Result* parameter value as the D-START *Result* parameter value, and

c) If the abstract value of the *Result* parameter is not “accepted” enter the *IDLE* state, or
d) Else enter the *DIALOGUE* state.

5.3.11 **CPDLC-message Service Request**

5.3.11.1 Upon receipt of a CPDLC-message service request, if the CPDLC-ASE is in the *DIALOGUE* state, the CPDLC-ASE shall:

a) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter, or
b) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,
c) Invoke D-DATA request with the APDU-element as the D-DATA *User Data* parameter value, and
d) Remain in the *DIALOGUE* state.

5.3.12 **CPDLC-end Service Request**

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

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,
b) Invoke D-DATA request with the APDU-element as the D-DATA *User Data* parameter value, if provided,
c) Start timer t_{end}, and
d) Enter the *END* state.

5.3.13 **DSC-end Service Request**

5.3.13.1 Upon receipt of a DSC-end service request, if the CPDLC-air-ASE is in the *DIALOGUE* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the DSC-end service *CPDLC Message* parameter, if provided,
b) Invoke D-END request with the APDU-element as the D-END *User Data* parameter value, if provided,
c) Start timer t_{end}, and
d) Enter the *END* state.

5.3.14 **CPDLC-end Service Response**

5.3.14.1 Upon receipt of a CPDLC-end service response, if the CPDLC-air-ASE is in the *END* state, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,
b) Invoke D-END response with the following:
   1) The APDU as the D-END *User Data* parameter value; if provided by the CPDLC-air-user, and
   2) The CPDLC-end service *Result* parameter value as the D-END *Result* parameter value as follows:
      i) “accepted as “accepted”, and
      ii) “rejected” as “rejected”, and
c) If *Result* is “rejected” enter the *DIALOGUE* state, or
d) Else enter the *IDLE* state.

5.3.15 **DSC-end Service Response**

5.3.15.1 Upon receipt of a DSC-end service response, if the CPDLC-ground-ASE is in the *END* state, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the DSC-end service *CPDLC Message* parameter , if provided,

b) Invoke D-END response with the following:
   1) The send APDU-element as the D-END *User Data* parameter; if provided by the CPDLC-ground-user, and
   2) The DSC-end service *Result* parameter value as the D-END *Result* parameter value as follows:
      i) "accepted as "accepted", and
      ii) "rejected" as "rejected (permanent)", and

c) If *Result* is "rejected" enter the *DIALOGUE* state, or

d) Else enter the *IDLE* state.

5.3.16 CPDLC-forward Service Request

5.3.16.1 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 containing the *CPDLC Message* parameter as the Forward Message,

b) Invoke D-START request with the following:
   1) the CPDLC-forward service *Called ICAO Facility Designator* parameter value as the D-START *Called Peer ID* parameter value,
   2) the CPDLC-start service *Calling ICAO Facility Designator* parameter value as the D-START *Calling Peer ID* parameter value,
   3) the D-START *Quality of Service* parameters set as follows:
      i) if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START QOS *Routing Class*, or
      ii) The abstract value of "normal priority flight safety messages", as the D-START *QOS Priority* parameter value, and
      iii) The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and
   4) the APDU as the D-START *User Data* parameter value;

c) Start timer t_{start}, and

d) Enter the *FORWARD* state.

5.3.17 CPDLC-user-abort Service Request

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

a) Stop any timer,

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

c) Enter the *IDLE* state.

5.3.18 D-ABORT Indication

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

a) Stop any timer,

b) If the CPDLC-user is an active user, then
   1) If the D-ABORT *Originator* parameter contains the abstract value "user" invoke CPDLC-user-abort service indication,
   2) Else, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided, and

c) Enter the *IDLE* state.

5.3.19 D-P-ABORT Indication

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

a) Stop any timer,
b) If the CPDLC-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”, and
c) Enter the *IDLE* state.

5.4 Exception Handling

5.4.1 A Timer Expires

5.4.1.1 If a CPDLC-ASE detects that a timer has expired, that ~~If a timer expires the~~ CPDLC-ASE shall:

a) Interrupt any current activity,
b) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with a CPDLCAbortReason [timer-expired] APDU message element,
c) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with a CPDLCAbortReason [timer-expired] APDU message element,
d) invoke D-ABORT request with:
 1) the abstract value “provider” as the D-ABORT *Originator* parameter value, and
 2) the APDU as the D-ABORT *User Data* parameter value, and
e) If the CPDLC-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “timer-expired”~~APDU~~ as the CPDLC-provider abort service *Reason* parameter value.

5.4.2 Unrecoverable System Error

5.4.2.1 If a CPDLC-ASE has an unrecoverable system error, the CPDLC-ASE shall:

a) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with a CPDLCAbortReason [undefined-error] APDU message element,
b) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with a CPDLCAbortReason [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-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “undefined-error”~~APDU~~ as the CPDLC-provider abort service *Reason* parameter value.

5.4.3 Invalid PDU

5.4.3.1 If the *User Data* parameter of a D-END confirmation with *Result* parameter set to the abstract value “rejected”, a D-START indication, a D-DATA indication, or a D-END indication, does not contain a valid PDU, the CPDLC-ASE shall:

a) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with a CPDLCAbortReason [invalid-PDU] APDU message element,
b) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with a CPDLCAbortReason [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-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “invalid-PDU”~~APDU~~ as the CPDLC-provider abort service *Reason* parameter value.

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 and the CPDLC-user is an active user, then the CPDLC-ASE shall invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value “invalid-PDU”.

5.4.4 Not Permitted PDU

5.4.4.1 If the *User Data* parameter of a D-START indication or D-DATA indication is a valid PDU, but is not a permitted PDU as defined within section 5.3, the CPDLC-ASE shall:

a) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with a CPDLCAbortReason [not-permitted-PDU] APDU message element,
b) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with a CPDLCAbortReason [not-permitted-PDU] APDU message element,
c) Invoke D-ABORT request with:
    1) the abstract value “provider” as the D-ABORT *Originator* parameter value, and
    2) the APDU as the D-ABORT *User Data* parameter value, and
d) If the CPDLC-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU”~~APDU~~ as the CPDLC-provider abort service *Reason* parameter value.

5.4.4.2 If the *User Data* parameter of a D-START confirmation is a valid PDU, but is not a permitted PDU as defined within section 5.3, the CPDLC-ASE shall:

a) If the D-START *Result* parameter is set to the abstract value “accepted”, then
    1) If the CPDLC-ASE is a CPDLC-air-ASE, create an AircraftPDUs APDU with a CPDLCAbortReason [not-permitted-PDU] APDU message element,
    2) If the CPDLC-ASE is a CPDLC-ground-ASE, create a GroundPDUs APDU with a CPDLCAbortReason [not-permitted-PDU] APDU message element,
    3) ~~If the D-START *Result* parameter is set to the abstract value “accepted”,~~ I~~i~~nvoke D-ABORT request with:
        i) the abstract value “provider” as the D-ABORT *Originator* parameter value, and
        ii) the APDU as the D-ABORT *User Data* parameter value, and
b) If the CPDLC-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU”~~APDU~~ as the CPDLC-provider-abort service *Reason* parameter value.

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

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”, and if the CPDLC-user is an active user, the CPDLC-ASE shall invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value “communication-service-error”.

5.5 CPDLC ASE State Tables

5.5.1 Introduction

Note 1: — This section defines the state tables for the CPDLC-air-ASE and the CPDLC-ground-ASE.

Note 2: — If the state tables shown in this section conflict with the textual statements made elsewhere in this SARPs, the textual statements take precedence.

5.5.2 CPDLC-ASE State Table

Note: — Table A-1 reflects the possible states of the CPDLC-ASE.

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**	**FORWARD**
Dialogue Service Events						
D-START Indication	• If mode is “dsc” then, DSC-start indication, set DSC=“true” else, CPDLC-start indication ⇒START-IND	cannot occur	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* “rejected (permanent)” and *Reject Source* “DS user”	cannot occur	• Stop timer t_{start} • If DSC=“true” then DSC-start confirmation, else CPDLC-start confirmation ⇒*IDLE*	cannot occur	cannot occur	cannot occur	• Stop timer t_{start} • CPDLC-forward confirmation ⇒*IDLE*
D-START Confirmation *Result* “accepted”	cannot occur	• Stop timer t_{start} • If DSC=“true” then DSC-start confirmation, else CPDLC-start confirmation ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur	• Stop timer t_{start} • if active user, CPDLC-provider-abort indication • D-ABORT request ⇒*IDLE*
D-DATA Indication	cannot occur	cannot occur	cannot occur	• CPDLC-message indication ⇒*DIALOGUE*	• CPDLC-message indication ⇒*END*	cannot occur
D-END Indication: CPDLC-ground-ASE only	cannot occur	cannot occur	cannot occur	• DSC-end indication ⇒*END*	cannot occur	cannot occur
D-END Indication: CPDLC-air-ASE only	cannot occur	cannot occur	cannot occur	• CPDLC-end indication ⇒*END*	cannot occur	cannot occur
D-END Confirmation: CPDLC-ground-ASE only *Result* “rejected (permanent)” and *Reject Source* “DS user”	cannot occur	cannot occur	cannot occur	cannot occur	• Stop timer t_{end} • CPDLC-end confirmation ⇒*DIALOGUE*	cannot occur

D-END Confirmation: CPDLC-air-ASE only *Result* “rejected (permanent)” and *Reject Source* “DS user”	cannot occur	cannot occur	cannot occur	cannot occur	• Stop timer t_{end} • DSC-end confirmation ⇒*DIALOGUE*	cannot occur
D-END Confirmation: CPDLC-ground-ASE only *Result* “accepted”	cannot occur	cannot occur	cannot occur	cannot occur	• Stop timer t_{end} • CPDLC-end confirmation ⇒*IDLE*	cannot occur
D-END Confirmation: CPDLC-air-ASE only *Result* “accepted”	cannot occur	cannot occur	cannot occur	cannot occur	• Stop timer t_{end} • DSC-end confirmation ⇒*IDLE*	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 *Result* “rejected”	not permitted	not permitted	• D-START response ⇒*IDLE*	not permitted	not permitted	not permitted
CPDLC-start Response *Result* “accepted”	not permitted	not permitted	• D-START response ⇒*DIALOGUE*	not permitted	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	not permitted
DSC-start Response *Result* “rejected”	not permitted	not permitted	• D-START response ⇒*IDLE*	not permitted	not permitted	not permitted
DSC-start Response *Result* “accepted”	not permitted	not permitted	• D-START response ⇒*DIALOGUE*	not permitted	not permitted	not permitted
CPDLC-message Request	not permitted	not permitted	not permitted	• D-DATA request ⇒*DIALOGUE*	not permitted	not permitted

CPDLC-end Request: CPDLC-ground-user only	not permitted	not permitted	not permitted	• D-END request • Start timer t_{end} ⇒*END*	not permitted	not permitted
CPDLC-end Service Response CPDLC-air-user only *Result* “rejected”	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*DIALOGUE*	not permitted
CPDLC-end Service Response CPDLC-air-user only *Result* “accepted”	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*IDLE*	not permitted
DSC-end Request: CPDLC-air-user only	not permitted	not permitted	not permitted	• D-END request • Start timer t_{end} ⇒*END*	not permitted	not permitted
DSC-end Service Response CPDLC-ground-user only *Result* “rejected”	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*DIALOGUE*	not permitted
DSC-end Service Response CPDLC-ground-user only *Result* “accepted”	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*IDLE*	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 ⇒*IDLE*	• Stop timer t_{start}, if set • D-ABORT request ⇒*IDLE*	• D-ABORT request ⇒*IDLE*	• Stop timer t_{end}; if set • D-ABORT request ⇒*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 ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-provider-abort indication ⇒*IDLE*	• If active user: CPDLC-provider-abort indication ⇒*IDLE*	• Stop timer t_{end}; if set • If active user: CPDLC-provider-abort indication ⇒*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 ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication ⇒*IDLE*	• If active user: CPDLC-user-abort indication ⇒*IDLE*	• Stop timer t_{end}; if set • If active user: CPDLC-user-abort indication ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication ⇒*IDLE*

D-P-ABORT indication	cannot occur	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication ⇒*IDLE*	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication ⇒*IDLE*	• If active user: CPDLC-provider-abort indication ⇒*IDLE*	• Stop timer t_{end}; if set • If active user: CPDLC-provider-abort indication ⇒*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 ⇒*IDLE*	cannot occur	cannot occur	cannot occur	• D-ABORT request • CPDLC-provider-abort indication ⇒*IDLE*
T_{end} Expires	cannot occur	cannot occur	cannot occur	cannot occur	• D-ABORT request • CPDLC-provider-abort indication ⇒*IDLE*	cannot occur

Table 5-2: CPDLC-ASE State Table

6. COMMUNICATION REQUIREMENTS

6.1 Encoding Rules

6.1.1 The CPDLC application shall use PER as defined in reference [2], using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in chapter 4 of this document, or a functionally equivalent means which provides the same result.

6.2 Dialogue Service Requirements

6.2.1 Primitive Requirements

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 chapter 5 of this document, the CPDLC-ground-ASE and the CPDLC-air-ASE shall exhibit external behavior consistent with the dialogue service, as described in reference [4], having been implemented and its primitives invoked.

6.2.2 Quality-of-Service Requirements

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

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

Note 1. — The application service RER for CPDLC is required to be 10^{-7} or better.

Note 2: — The RER takes into account non-delivery for messages over a given dialogue.

~~Routing Policy~~

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 6-1.

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

Table 6-1: Mapping Between Class of Communication and Routing Class Abstract Values

Note: — ATSC values are defined in [5].

6.3 CPDLC AE Control Functions Requirements

6.3.1 The title of the CPDLC application used as the AE-Qualifier for the CPDLC application shall be the IA5 character string “CPC”.

7. CPDLC USER REQUIREMENTS

7.1 Introduction

Note 1: — Requirements imposed on CPDLC-uses~~r~~ concerning CPDLC messages and interfacing with the CPDLC-ASEs are presented in this chapter.

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

Note 3: — A CPDLC message is:

a) what the CPDLC-user provides the CPDLC-service as the CPDLC Message or Reject Reason parameter, when invoking a CPDLC service request or response primitive, or

b) what the CPDLC-user receives in the same parameters from the CPDLC service indication or confirmation primitives.

Note 4: — In this chapter the terms CPDLC message, message, uplink message and downlink message are used interchangeably, and equate to a CPDLC message. When the terms "send" and "transmit" are used this means that the CPDLC-user has invoked a CPDLC service request or response primitive. When the term "receive" is used this means that a CPDLC indication or confirmation primitive parameter containing a CPDLC message has been provided by the CPDLC service.

7.2 General

7.2.1 General CPDLC Service Requirements

7.2.1.1 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.

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

7.2.2 General Parameter Requirements

7.2.3 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 shall set the *Class of Communication Service* parameter.

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

7.3 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 require 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.

7.3.1 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.

7.3.2 For each CPDLC message the CPDLC-user sends ground/ground it shall provide the following information:

a) the aircraft identification to which the ground/ground message refers,
b) date and time,
c) from one to five message element identifiers, and
d) data as required for each message element identification included.

7.3.3 Message Identification Number

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

7.3.3.2 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.

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

7.3.3.4 If the CPDLC-user sends a message containing the ERROR message element instead of the expected response message, the ERROR message shall contain the received message identification number as the message reference number.

7.3.4 Error Conditions

7.3.4.1 No Available Message Identification Numbers

7.3.4.1.1 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 CPDLC a message containing the ERROR [errorinformation] message element with the value [no-message-identification-numbers-available].value as the *Reason* parameter.

7.4 CPDLC Message Receipt Requirements

7.4.1 Urgency Requirements

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

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

7.4.2 Alerting Requirements

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

7.4.3 CPDLC/DSC Distinction

7.4.3.1 Upon receipt of a CPDLC message the CPDLC-user shall provide a distinction between CPDLC messages received from the Current Data Authority and those received from a Downstream Data Authority.

7.4.4 Air/Ground - Ground/Ground Distinction

7.4.4.1 Upon receipt of a CPDLC message the CPDLC-user shall provide a distinction between CPDLC messages received from an aircraft and those received from another ground system.

7.4.5 Logical Acknowledgment Prohibited

7.4.5.1 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.

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

7.4.6 Message Reference Numbers

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

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

7.4.7 Message Response Requirements

Note 1. — A message sequence initiated by data link should be closed by data link.

Note 2. — If a message sequence exchange initiated by data link is subsequently closed by voice, local procedures must be in place to ensure deletion of outstanding data link messages requiring closure.

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

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

7.4.7.3 If a message is received that requires a response, the CPDLC-user shall either:

a) send any permitted response messages and then send a closure response message, or
b) send a closure response message.

7.4.7.4 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.

7.4.7.5 When a message is received by the CPDLC-user requiring a logical acknowledgment response, the CPDLC-user shall respond with either a CPDLC message containing only a LOGICAL ACKNOWLEDGEMENT message element, or a message containing an ERROR message element.

7.4.7.6 A logical acknowledgment response message, if required, shall be sent prior to sending any other related response message(s), except a response message containing an ERROR message element.

7.4.7.7 When the CPDLC-air-user receives a message with a W/U RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, WILCO, UNABLE, or ERROR message element.

7.4.7.8 When the CPDLC-air-user receives a message with a W/U RESP attribute, the closure response message shall contain at least a WILCO, UNABLE, or ERROR message element.

7.4.7.9 When the CPDLC-air-user receives a message with a A/N RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, AFFIRM, NEGATIVE, or ERROR message element.

7.4.7.10 When the CPDLC-air-user receives a message with a A/N RESP attribute, the closure response message shall contain at least a AFFIRM, NEGATIVE, or ERROR message element.

7.4.7.11 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, or ERROR message element.

7.4.7.12 When the CPDLC-air-user receives a message with a R RESP attribute, the closure response message shall contain at least a ROGER or ERROR message element.

7.4.7.13 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.

7.4.7.14 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 ACKNOWLEDGEMENT shall constitute the closure response message.

7.4.7.15 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.

7.4.7.16 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 ACKNOWLEDGEMENT message element shall constitute the closure response message.

7.4.7.17 When the CPDLC-air-user receives a message with a N RESP attribute, the only permitted response shall be a closure response message that contains a LOGICAL ACKNOWLEDGMENT or ERROR message element, if a logical acknowledgment is required.

7.4.7.18 When the CPDLC-ground-user receives an air/ground message with a N RESP attribute, the only permitted response shall be a closure response message that contains a LOGICAL ACKNOWLEDGMENT or ERROR message element, if a logical acknowledgment is required.

7.4.7.19 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.

7.4.8 Error Conditions

7.4.8.1 Duplicate Message Identification Numbers

7.4.8.1.1 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 service with a CPDLC message containing the CPDLCAbortReason with value [duplicate-message-identification-number] value as the *Reason* parameter.

7.4.8.2 Invalid Reference Number

7.4.8.2.1 If the CPDLC-user receives a message containing a message reference number which is not identical to any message identification number currently in use, the CPDLC-user shall:

a) invoke CPDLC-message request with the ERROR [errorinformation] message element with the [unrecognizedMsgReferenceNumber] value as the *CPDLC Message* parameter, and

b) disregard the received message.

7.5 CPDLC-air-user Requirements

7.5.1 The CPDLC-start Service

7.5.1.1 Invoking the CPDLC-start request

7.5.1.1.1 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.

7.5.1.1.2 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.

7.5.1.1.3 If a CPDLC-air-user has invoked a CPDLC-start request, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive pertaining to the ground system addressed in the CPDLC-start service, except the CPDLC-user-abort request, until after it has received a CPDLC-start confirmation.

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

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

7.5.1.2.2 Upon receipt of a CPDLC-start indication the CPDLC-air-user shall invoke the CPDLC-start response, with the response parameters set as follows:

a) The *Result* parameter to the abstract value of “accepted” if:
    1) There is no existing CPDLC service, or
    2) CPDLC service exists and the request is from either the Current Data Authority or Next Data Authority,
b) Else set the *Result* parameter is set to the abstract value “rejected” and the *Reject Reason* to a CPDLC message with the message element NOT AUTHORIZED NEXT DATA AUTHORITY.

7.5.1.2.3 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.

7.5.1.2.4 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value “rejected” any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be disregarded.

7.5.1.2.5 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” and the request is from the Current Data Authority any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be processed.

7.5.1.2.6 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” and the request is from the Next Data Authority any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be disregarded.

7.5.1.2.7 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” the CPDLC-air-user shall:

a) Establish an association between a CPDLC-ASE invocation and a ground system ICAO facility designator 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 ICAO facility designator contained CPDLC-start indication *Calling Peer Identifier* parameter is the Next Data Authority associate the CPDLC-ASE invocation with the Next Data Authority.

7.5.1.2.8 If CPDLC-start indication has been received, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive with this ground system, except the CPDLC-user-abort request, until after it has invoked the CPDLC-start response.

7.5.1.3 Receipt of a CPDLC-start confirmation

7.5.1.3.1 If a CPDLC-start confirmation has been received with a *Result* parameter containing the abstract value "accepted" the CPDLC-air-user shall:

a) Establish an association between a CPDLC-ASE invocation and a ground system ICAO facility designator contained in CPDLC-start request *Called Peer Identifier* parameter,

b) If there is no Current Data Authority, associate the CPDLC-ASE invocation with the Current Data Authority, or

c) If the ICAO facility designator contained CPDLC-start indication *Called Peer Identifier* parameter is the Next Data Authority associate the CPDLC-ASE invocation with the Next Data Authority.

7.5.2 The DSC-start Service

7.5.2.1 Invoking the DSC-start request

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

7.5.2.1.2 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.

7.5.2.1.3 If a CPDLC-air-user has invoked a DSC-start request, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive with this ground system, except the CPDLC-user-abort request, until after it has received a DSC-start confirmation.

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

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

7.5.2.2.2 Upon receipt of a DSC-start indication, the CPDLC-ground-user shall invoke a DSC-start response, with the response parameters set as follows:

a) the *Result* parameter set to the abstract value "accepted" or "rejected", and

b) if, and only if, the *Result* parameter is set to the abstract value "rejected", then set the *Reject Reason* parameter to a CPDLC message with the message element SERVICE UNAVAILABLE.

7.5.2.3 Receipt of a DSC-start confirmation

7.5.2.3.1 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 ICAO facility designator contained in DSC-start request *ICAO Facility Designator* parameter,

b) Associate the CPDLC-ASE invocation with a Downstream Data Authority.

7.5.3 The CPDLC-message Service

7.5.3.1 Receipt of a CPDLC-message Indication

7.5.3.1.1 Upon receipt of a CPDLC-message indication, if the indication is from the Current Data Authority or a Downstream Data Authority the CPDLC-air-user shall process the CPDLC message contained in the *CPDLC Message* parameter.

7.5.3.1.2 If a CPDLC-message indication is received from the Current Data Authority containing at least the uplink message element NEXT DATA AUTHORITY, and having no more than one NEXT DATA AUTHORITY message elements, indicating that the specified ICAO facility designator is 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 CPDLCAbortReason 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.

7.5.3.1.3 If a CPDLC-message indication is received from the Current Data Authority containing more than one NEXT DATA AUTHORITY uplink message elements, the CPDLC-air-user shall disregard the Next Data Authority designations, and invoke CPDLC-message service request with the ERROR [errorinformation] message element with the [moreThanOneNextDataAuthorityElement] value as the *CPDLC Message* parameter value.

7.5.3.1.4 If a CPDLC-message indication is received containing at least the uplink message element NEXT DATA AUTHORITY, and it is not from the Current Data Authority the message shall be disregarded.

7.5.3.1.5 Upon receipt of a CPDLC-message indication, if the indication is not from the Current Data Authority or a Downstream Data Authority, the CPDLC-air-user shall:

a) Invoke CPDLC-message service request with the *CPDLC Message* parameter containing a message with the message element NOT CURRENT DATA AUTHORITY, and
b) Disregard the received message contained in the CPDLC-message indication *CPDLC Message* parameter.

7.5.4 The CPDLC-end Service

7.5.4.1 The CPDLC-end Service Request

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

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

7.5.4.2.1 Upon receipt of a CPDLC-end indication, the CPDLC-air-user shall invoke a CPDLC-end response within 0.5 seconds.

7.5.4.2.2 If a CPDLC-end indication is received but it is not from the Current Data Authority the CPDLC-air-user shall:

a) Invoke CPDLC-end response with
   1) the *CPDLC Message* parameter containing a message with the message element NOT CURRENT DATA AUTHORITY, and
   2) the *Result* parameter set to the abstract value “rejected”, and
b) Disregard any message provided in the CPDLC-end indication *CPDLC Message* parameter.

7.5.4.2.3 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has sent a message that requires a response for which it has not received a closure response then the CPDLC-air-user shall:

a) Invoke CPDLC-end response with:
   1) the *CPDLC Message* parameter containing a CPDLC ATCDownlinkMessage with the ERROR [errorinformation] message element with the [endServiceWithPendingMsgs] value, and
   2) the *Result* parameter set to the abstract value “rejected”, and
b) Disregard any message provided in the CPDLC-end indication *CPDLC Message* parameter.

7.5.4.2.4 If a CPDLC-end indication is received from the Current Data Authority and the CPDLC-air-user has received a message for which a response is required, and it has not yet sent the closure response to that message then the CPDLC-air-user shall:

a) Invoke CPDLC-end response with:
   1) the *CPDLC Message* parameter containing a CPDLC ATCDownlinkMessage with the ERROR [errorinformation] message element with the [endServiceWithPendingMsgs] value, and
   2) the *Result* parameter set to the abstract value “rejected”, and
b) Disregard any message provided in the CPDLC-end indication *CPDLC Message* parameter.

7.5.4.2.5 If a CPDLC-end indication is received that is from the Current Data Authority and the *CPDLC Message* parameter contains a message requiring a response, the CPDLC-air-user shall:

a) If responding with any permitted CPDLC message response that is not the closure response, invoke CPDLC-message request with the response message as the CPDLC Message parameter, then
b) If desired, invoke CPDLC-message request with the closure response message as the CPDLC Message parameter, then
c) Invoke CPDLC-end response with:
   1) the CPDLC Message parameter as the CPDLC closure response if not already sent, and

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

7.5.4.2.6 If a CPDLC-end indication is received that is from the Current Data Authority without a *CPDLC Message* parameter, the CPDLC-air-user shall invoke CPDLC-end response with:

a) the *CPDLC Message* parameter with a CPDLC message if desired, and
b) the *Result* parameter set to the abstract value “rejected” or “accepted”.

7.5.4.2.7 Upon invoking CPDLC-end response with *Result* parameter set to “accepted”, the CPDLC-air-user shall:

a) delete any association with a ground system and Current Data Authority, and
b) if a ground system is designated as Next Data Authority and an association with a CPDLC-ASE exists, replace the Next Data Authority association with a Current Data Authority association, or
c) if a ground system is designated as Next Data Authority and no association with a CPDLC-ASE exists, delete Next Data Authority association with any ground system.

7.5.4.2.8 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.

7.5.5 The DSC-end Service

7.5.5.1 The DSC-end Request

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

7.5.5.2 Receipt of a DSC-end Indication and Invoking a DSC-end Response

7.5.5.2.1 Upon receipt of a DSC-end service indication, the CPDLC-ground-user shall invoke a DSC-end service response within 0.5 seconds.

7.5.5.2.2 If a DSC-end indication is received and the CPDLC-ground-user has sent a message that requires a response for which it has not received a closure response then the CPDLC-ground-user shall:

a) Invoke DSC-end response with:
   1) the *CPDLC Message* parameter containing a CPDLC ATCDownlinkMessage with the ERROR [errorinformation] message element with the [endServiceWithPendingMsgs] value, and
   2) the *Result* parameter set to the abstract value “rejected”, and
b) Disregard any message provided in the DSC-end indication *CPDLC Message* parameter.

7.5.5.2.3 If a DSC-end indication is received and the CPDLC-ground-user has received a message for which a response is required, and it has not yet sent the closure response to that message then the CPDLC-ground-user shall:

a) Invoke DSC-end response with:
   1) the *CPDLC Message* parameter containing a CPDLC ATCDownlinkMessage with the ERROR [errorinformation] message element with the [endServiceWithPendingMsgs] value, and
   2) the *Result* parameter set to the abstract value “rejected”, and
b) Disregard any message provided in the DSC-end indication *CPDLC Message* parameter.

7.5.5.2.4 If a DSC-end indication is received and the *CPDLC Message* parameter contains a message requiring a response, the CPDLC-ground-user shall:

a) If responding with any permitted CPDLC message response that is not the closure response, invoke CPDLC-message request with the response message as the *CPDLC Message* parameter, then
b) If desired, invoke CPDLC-message request with the closure response message as the *CPDLC Message* parameter, then
c) Invoke DSC-end response with:
   1) the *CPDLC Message* parameter as the CPDLC closure response if not already sent, and
   2) the *Result* parameter set to the abstract value “rejected” or “accepted”.

7.5.5.2.5 If a DSC-end indication is received without a *CPDLC Message* parameter, the CPDLC-ground-user shall invoke DSC-end response with:

a) the *CPDLC Message* parameter with a CPDLC message if desired, and
b) the *Result* parameter set to the abstract value “rejected” or “accepted”.

7.5.6 The CPDLC-user-abort Service

7.5.6.1 Receipt of a CPDLC-abort Indication

7.5.6.1.1 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 abstract value " current data authority abort", and
c) Delete any association of a ground system to a Next Data Authority.

7.5.6.1.2 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.

7.6 CPDLC-Ground-User Requirements

7.6.1 The CPDLC-start Service

7.6.1.1 Invoking the CPDLC-start request

7.6.1.1.1 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.

7.6.1.1.2 If a CPDLC-ground-user has invoked a CPDLC-start request, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive, except the CPDLC-user-abort request with that aircraft, until after it has received a CPDLC-start confirmation.

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

7.6.1.2.1 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.

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

7.6.1.2.3 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value "rejected" then the *Reject Reason* shall be an uplink message containing only the SERVICE UNAVAILABLE message element.

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

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

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

7.6.1.2.7 If CPDLC-start indication has been received, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive, except the CPDLC-user-abort request with that aircraft, until after it has invoked the CPDLC-start response.

7.6.1.3 Receipt of a CPDLC-start confirmation

7.6.1.3.1 If a CPDLC-start confirmation has been received with a *Result* parameter containing the abstract value "accepted" the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in CPDLC-start request *Called Peer Identifier* parameter.

7.6.2 The DSC-start Service

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

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

7.6.2.1.2 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.

7.6.2.1.3 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value "rejected" then the *Reject Reason* shall be an uplink message containing only the SERVICE UNAVAILABLE message element.

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

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

7.6.2.1.6 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 Identifier* parameter.

7.6.2.1.7 If DSC-start indication has been received, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive, except the CPDLC-user-abort request, until after it has invoked the DSC-start response.

7.6.3 The CPDLC-end Service

7.6.3.1 The CPDLC-end Request

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

7.6.3.1.2 The CPDLC-ground-user shall be prohibited from invoking the CPDLC-end request if:

a) it has sent a message that requires a response for which it has not received a closure response, or
b) it has received a message for which a response is required, and it has not yet sent the closure response.

7.6.4 The DSC-end Service

7.6.4.1 Receipt of a DSC-end Indication and Invoking DSC-end Response

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

7.6.4.1.2 If DSC-end 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 DSC-end response.

7.6.5 The CPDLC-forward Service

7.6.5.1 Invoking the CPDLC-forward request

7.6.5.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-forward request.

7.7 Message Intent

7.7.1 Purpose

Note: — This section 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 section 4 of the SARPs; however, section 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 within this section are one possible means of presenting the information. These recommendations are generally consistent with current ICAO practices for displaying ATC information.

7.7.2 Message elements shall comply with the intent and use as presented in the following Tables in section 7.6.

7.7.3 Up-Link Messages

Note: — Uplink messages for CPDLC are presented in this section.

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
0	Indicates that ATS cannot comply with the request.	UNABLE	N	M	N
1	Indicates that ATS has received the request and will respond shortly.	STANDBY	N	L	N
2	Indicates that ATS has received the request but it has been deferred until later.	REQUEST DEFERRED	N	L	N
3	Indicates that ATS has received and understood the request.	ROGER	N	L	N
4	Yes.	AFFIRM	N	L	N
5	No.	NEGATIVE	N	L	N
235	Notification of receipt of unlawful interference message.	ROGER 7500	U	H	N
211	Indicates that the ATS has received the request and has passed it to the Next Control Authority.	REQUEST FORWARDED	N	L	N
218	Indicates to the pilot that the request has already been received on the ground.	REQUEST ALREADY RECEIVED	L	N	N

Table 7-1: Responses/Acknowledgments (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Notification that an altitude change instruction should be expected.	EXPECT [altitude]	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
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 altitude.	AT [time] EXPECT CLIMB TO [altitude]	L	L	R
14	Notification that an instruction should be expected for the aircraft to commence climb at the specified position to the specified altitude.	AT [position] EXPECT CLIMB TO [altitude]	L	L	R
15	Notification that an instruction should be expected for the aircraft to commence descent at the specified time to the specified altitude.	AT [time] EXPECT DESCENT TO [altitude]	L	L	R
16	Notification that an instruction should be expected for the aircraft to commence descent at the specified position to the specified altitude.	AT [position] EXPECT DESCENT TO [altitude]	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 altitude.	AT [time] EXPECT CRUISE CLIMB TO [altitude]	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 altitude.	AT [position] EXPECT CRUISE CLIMB TO [altitude]	L	L	R
19	Instruction to maintain the specified altitude.	MAINTAIN [altitude]	N	M	W/U
20	Combined instruction that a climb to a specified altitude is to commence and the altitude is to be maintained when reached.	CLIMB TO AND MAINTAIN [altitude]	N	M	W/U
21	Combined instruction that at the specified time, a climb to the specified altitude is to commence and once reached the specified altitude is to be maintained.	AT [time] CLIMB TO AND MAINTAIN [altitude]	N	M	W/U

22	Combined instruction that at the specified position, a climb to the specified altitude is to commence and once reached the specified altitude is to be maintained.	AT [position] CLIMB TO AND MAINTAIN [altitude]	N	M	W/U
185	Combined instruction that after passing the specified position, a climb to the specified altitude is to commence and once reached the specified altitude is to be maintained.	AFTER PASSING [position] CLIMB TO AND MAINTAIN [altitude]	N	M	W/U
23	Combined instruction that a descent to a specified altitude is to commence and the altitude is to be maintained when reached.	DESCEND TO AND MAINTAIN [altitude]	N	M	W/U
24	Combined instruction that at a specified time a descent to a specified altitude is to commence and once reached the specified altitude is to be maintained.	AT [time] DESCEND TO AND MAINTAIN [altitude]	N	M	W/U
25	Combined instruction that at the specified position a descent to the specified altitude is to commence and when the specified altitude is reached it is to be maintained.	AT [position] DESCEND TO AND MAINTAIN [altitude]	N	M	W/U
186	Combined instruction that after passing the specified position, a descent to the specified altitude is to commence and once reached the specified altitude is to be maintained.	AFTER PASSING [position] DESCEND TO AND MAINTAIN [altitude]	N	M	W/U
26	Combined instruction that a climb is to commence at a rate such that the specified altitude is reached at or before the specified time.	CLIMB TO REACH [altitude] BY [time]	N	M	W/U
27	Combined instruction that a climb is to commence at a rate such that the specified altitude is reached at or before the specified position.	CLIMB TO REACH [altitude] BY [position]	N	M	W/U
28	Combined instruction that a descent is to commence at a rate such that the specified altitude is reached at or before the specified time.	DESCEND TO REACH [altitude] BY [time]	N	M	W/U
29	Combined instruction that a descent is to commence at a rate such that the specified altitude is reached at or before the specified position.	DESCEND TO REACH [altitude] BY [position]	N	M	W/U
192~~7~~	Combined instruction that a change of altitude is to continue, but at a rate such that the specified altitude is reached at or before the specified time.~~Combined instruction that a change of altitude is to commence at a rate such that the specified altitude is reached at or before the specified time.~~	REACH [altitude] BY [time]	N	M	W/U

209	Combined instruction that a change of altitude is to continue, but at a rate such that the specified altitude is reached at or before the specified position~~Combined instruction that a change of altitude is to commence at a rate such that the specified altitude is reached at or before the specified position.~~	REACH [altitude] BY [position]	N	M	W/U
30	An altitude within the defined block altitude specified is to be maintained.	MAINTAIN BLOCK [altitude] TO [altitude]	N	M	W/U
31	Combined instruction that a climb to an altitude within the block altitude defined is to commence.	CLIMB TO AND MAINTAIN BLOCK [altitude] TO [altitude]	N	M	W/U
32	Combined instruction that a descent to an altitude within the block altitude defined is to commence.	DESCEND TO AND MAINTAIN BLOCK [altitude] TO [altitude]	N	M	W/U
34	A cruise climb is to commence and continue until the specified altitude is reached.	CRUISE CLIMB TO [altitude]	N	M	W/U
35	A cruise climb can commence once above the specified altitude.	CRUISE CLIMB ABOVE [altitude]	N	M	W/U
219	Instruction to stop the climb below the previously assigned altitude.	STOP CLIMB AT [altitude]	U	M	W/U
220	Instruction to stop the descent above the previously assigned altitude.	STOP DESCENT AT [altitude]	U	M	W/U
36	The climb to the specified altitude should be made at the aircraft's best rate.	EXPEDITE CLIMB TO [altitude]	U	M	W/U
37	The descent to the specified altitude should be made at the aircraft's best rate.	EXPEDITE DESCENT TO [altitude]	U	M	W/U
38	Urgent instruction to immediately climb to the specified altitude.	IMMEDIATELY CLIMB TO [altitude]	D	H	W/U
39	Urgent instruction to immediately descend to the specified altitude.	IMMEDIATELY DESCEND TO [altitude]	D	H	W/U
40	Urgent instruction to immediately stop a climb once the specified altitude is reached.	IMMEDIATELY STOP CLIMB AT [altitude]	D	H	W/U
41	Urgent instruction to immediately stop a descent once the specified altitude is reached.	IMMEDIATELY STOP DESCENT AT [altitude]	D	H	W/U
171	Instruction to climb at not less than the specified rate.	CLIMB AT [vertical rate] MINIMUM	N	M	W/U
172	Instruction to climb at not above the specified rate.	CLIMB AT [vertical rate] MAXIMUM	N	M	W/U
173	Instruction to descend at not less than the specified rate.	DESCEND AT [vertical rate] MINIMUM	N	M	W/U
174	Instruction to descend at not above the specified rate.	DESCEND AT [vertical rate] MAXIMUM	N	M	W/U
236	Authorization for the pilot to conduct flight at any altitude from the minimum IFR altitude up to and including the altitude specified.	CRUISE [altitude]	N	M	W/U

Table 7-2: Vertical Clearances (uplink)

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

56	The specified position is to be crossed at a speed equal to or less than the specified speed and the specified speed or less is to be maintained until further advised.	CROSS [position] AT OR LESS THAN [speed]	N	M	W/U
57	The specified position is to be crossed at a speed equal to or greater than the specified speed and the specified speed or greater is to be maintained until further advised.	CROSS [position] AT OR GREATER THAN [speed]	N	M	W/U
58	The specified position is to be crossed at the specified time and the specified altitude.	CROSS [position] AT [time] AT [altitude]	N	M	W/U
59	The specified position is to be crossed at or before the specified time and at the specified altitude.	CROSS [position] AT OR BEFORE [time] AT [altitude]	N	M	W/U
60	The specified position is to be crossed at or after the specified time and at the specified altitude.	CROSS [position] AT OR AFTER [time] AT [altitude]	N	M	W/U
61	Combined instruction that the specified position is to be crossed at the specified altitude and speed and the altitude and speed are to be maintained.	CROSS [position] AT AND MAINTAIN [altitude] AT [speed]	N	M	W/U
62	Combined instruction that at the specified time the specified position is to be crossed at the specified altitude and the altitude is to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [altitude]	N	M	W/U
63	Combined instruction that at the specified time the specified position is to be crossed at the specified altitude and speed and the altitude and speed are to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [altitude] AT [speed]	N	M	W/U

Table 7-3: Crossing Constraints (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 [distance offset] [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 [distance offset] [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 [distance offset] [direction] OF ROUTE	N	M	W/U
67	The cleared flight route is to be rejoined.	PROCEED BACK ON ROUTE	N	M	W/U
68	The cleared flight route is to be rejoined at or before the specified position.	REJOIN ROUTE BY [position]	N	M	W/U
69	The cleared flight route is to be rejoined at or before the specified time.	REJOIN ROUTE BY [time]	N	M	W/U
70	Notification that a clearance may be issued to enable the aircraft to rejoin the cleared route at or before the specified position.	EXPECT BACK ON ROUTE BY [position]	L	L	R
71	Notification that a clearance may be issued to enable the aircraft to rejoin the cleared route at or before the specified time.	EXPECT BACK ON ROUTE BY [time]	L	L	R
72	Instruction to resume own navigation following a period of tracking or heading clearances. May be used in conjunction with an instruction on how or where to rejoin the cleared route.	RESUME OWN NAVIGATION	N	M	W/U

Table 7-4: Lateral Offsets (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 altitude, directly to the specified position.	AT [altitude] PROCEED DIRECT TO [position]	N	M	W/U
79	Instruction to proceed to the specified position via the specified route.	CLEARED TO [position] VIA [route clearance]	N	M	W/U
80	Instruction to proceed via the specified route.	CLEARED [route clearance]	N	M	W/U
81	Instruction to proceed in accordance with the specified procedure.	CLEARED [procedure name]	N	M	W/U
33	Instruction to proceed to the intended destination which has no instrument approach procedure authorized or available, and is in uncontrolled airspace.	CLEARED OUT OF 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 [distance offset] [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 altitude is reached may be issued.	AT [altitude] EXPECT DIRECT TO [position]	L	L	R
91	Instruction to enter a holding pattern with the specified characteristics at the specified position and altitude.	HOLD AT [position] MAINTAIN [altitude] 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 altitude.	HOLD AT [position] AS PUBLISHED MAINTAIN [altitude]	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 onto the specified heading.	TURN [direction] HEADING [degrees]	N	M	W/U
95	Instruction to turn left or right as specified onto the specified track.	TURN [direction] GROUND TRACK [degrees]	N	M	W/U
215	Instruction to turn a specified number of degrees left or right.	TURN [degrees][direction]	N	M	W/U
190	Instruction to fly on the specified heading.	FLY HEADING [degrees]	N	M	W/U
96	Instruction to continue to fly on the current heading.	CONTINUE PRESENT HEADING	N	M	W/U
97	Instruction to fly on the specified heading from the specified position.	AT [position] FLY HEADING [degrees]	N	M	W/U
221	Instruction to stop turn at the specified heading prior to reaching the previously assigned heading.	STOP TURN HEADING [degrees]	U	M	W/U
98	Instruction to turn immediately left or right as specified onto the specified heading.	IMMEDIATELY TURN [direction] HEADING [degrees]	D	H	W/U
99	Notification that a clearance may be issued for the aircraft to fly the specified procedure.	EXPECT [procedure name]	L	L	R

Table 7-5: Route Modifications (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 altitude.	AT [altitude] 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 altitude.	AT [altitude] EXPECT [speed] TO [speed]	L	L	R
106	The specified speed is to be maintained.	MAINTAIN [speed]	N	M	W/U
188	After passing the specified position the specified speed is to be maintained.	AFTER PASSING [position] MAINTAIN [speed]	N	M	W/U
107	The present speed is to be maintained.	MAINTAIN PRESENT SPEED	N	M	W/U
108	The specified speed or a greater speed is to be maintained.	MAINTAIN [speed] OR GREATER	N	M	W/U
109	The specified speed or a lesser speed is to be maintained.	MAINTAIN [speed] OR LESS	N	M	W/U
110	A speed with the specified range is to be maintained.	MAINTAIN [speed] TO [speed]	N	M	W/U
111	The present speed is to be increased to the specified speed and maintained until further advised.	INCREASE SPEED TO [speed]	N	M	W/U
112	The present speed is to be increased to the specified speed or greater, and maintained at or above the specified speed until further advised.	INCREASE SPEED TO [speed] OR GREATER	N	M	W/U
113	The present speed is to be reduced to the specified speed and maintained until further advised.	REDUCE SPEED TO [speed]	N	M	W/U
114	The present speed is to be reduced to the specified speed or less and maintained at or below the specified speed until further advised.	REDUCE SPEED TO [speed] OR LESS	N	M	W/U
115	The specified speed is not to be exceeded.	DO NOT EXCEED [speed]	N	M	W/U

116	Notification that the aircraft need no longer comply with the previously issued speed restriction.	RESUME NORMAL SPEED	N	M	W/U
189	The present speed is to be changed to the specified speed.	ADJUST SPEED TO [speed]	N	M	W/U
222	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~~L~~	W/U

Table 7-6: Speed Changes (uplink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
117	The ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	CONTACT [icaounitname] [frequency]	N	M	W/U
118	At the specified position the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [position] CONTACT [icaounitname] [frequency]	N	M	W/U
119	At the specified time the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [time] CONTACT [icaounitname] [frequency]	N	M	W/U
120	The ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	MONITOR [icaounitname] [frequency]	N	M	W/U
121	At the specified position the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [position] MONITOR [icaounitname] [frequency]	N	M	W/U
122	At the specified time the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [time] MONITOR [icaounitname] [frequency]	N	M	W/U
123	The specified beacon code (SSR code) is to be selected.	SQUAWK [beacon code]	N	M	W/U
124	The SSR transponder responses are to be disabled.	STOP SQUAWK	N	M	W/U
125	The SSR transponder responses should include altitude information.	SQUAWK MODE CHARLIE	N	M	W/U
126	The SSR transponder responses should no longer include altitude information.	STOP SQUAWK MODE CHARLIE	N	M	W/U
179	The ‘ident’ function on the SSR transponder is to be actuated.	SQUAWK IDENT	N	M	W/U

Table 7-7: Contact/Monitor/Surveillance 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	R
128	Instruction to report when the aircraft has left the specified altitude.	REPORT LEAVING [altitude]	N	L	R
129	Instruction to report when the aircraft is in altitude flight at the specified altitude.	REPORT WHEN LEVEL AT [altitude]	N	L	R
175	Instruction to report when the aircraft has reached the specified altitude.	REPORT REACHING [altitude]	N	L	R
180	Instruction to report when the aircraft is within the specified altitude range.	REPORT REACHING BLOCK [altitude] TO [altitude]	N	L	R
130	Instruction to report when the aircraft has passed the specified position.	REPORT PASSING [position]	N	L	R
181	Instruction to report the present distance to or from the specified position.	REPORT DISTANCE [to/from] [position]	N	M	Y
184	Instruction to report at the specified time the distance to or from the specified position.	AT TIME [time] REPORT DISTANCE [to/from] [position]	N	L	Y
228	Instruction to report the estimated time of arrival at the specified position	REPORT ETA [position]	L	L	Y
131	Instruction to report the amount of fuel remaining and the number of persons on board.	REPORT REMAINING FUEL AND PERSONS~~SOULS~~ ON BOARD	U	M	Y
132	Instruction to report the present position.	REPORT POSITION	N	M	Y
133	Instruction to report the present altitude.	REPORT PRESENT ALTITUDE	N	M	Y
134	Instruction to report the requested~~present~~ speed.	REPORT [speed qualifier] [speed type] SPEED	N	M	Y
135	Instruction to confirm and acknowledge the currently assigned altitude.	CONFIRM ASSIGNED ALTITUDE	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
144	Instruction to report the selected beacon 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 altitude.	STATE PREFERRED ALTITUDE~~LEVEL~~	L	L	Y
232	Instruction to indicate the pilot's preferred time and/or position to commence descent to the aerodrome of intended arrival.	STATE TOP OF DESCENT	L	L	Y

Table 7-8: Report/Confirmation Requests (uplink)

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

Table 7-9: Negotiation Requests (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.	[icao facility designator] 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	Indication that radar identification has been lost.	IDENTIFICATION LOST	N	M	R
157	A continuous transmission is detected on the specified frequency. Check the microphone button.	CHECK STUCK MICROPHONE [frequency]	U	M	N
158	ATS advisory that the ATIS information identified by the specified code is the current ATIS information.	ATIS [atis code]	N	L	R
212	ATS advisory that the specified ATIS information at the specified airport is current.	[icao facility designator] ATIS [atis code] CURRENT	N	L	R
214	ATS advisory that indicates the RVR value for the specified runway.	RUNWAY [runway] VISUAL RANGE [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 7-10: Air Traffic Advisories (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
159	A system generated message that the ground system has detected an error.	ERROR [error information]	U	M	N
160	Notification to the avionics that the next data authority is the specified ATSU.	NEXT DATA AUTHORITY [icao facility designator]	L	N	N
161	Notification to the avionics that the data link connection with the current data authority is being terminated.	END SERVICE	L	N	N
162	Notification that the ground system does not support this message.	SERVICE UNAVAILABLE	L	L	N
234	Notification that the ground system does not have a flight plan for that aircraft.	FLIGHT PLAN NOT HELD	L	L	N
163	Notification to the pilot of an ATSU identifier.	[icao facility designator]	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 7-11: System Management 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 ~~previous~~ communication should be ignored.	DISREGARD	U	M	R
176	Notification that the operator is responsible for maintaining separation from other traffic and is also responsible for maintaining Visual Meteorological Conditions.	MAINTAIN OWN SEPARATION AND VMC	N	M	W/U
177	Used in conjunction with a clearance/instruction to indicate that the operator may execute when prepared to do so.	AT PILOTS DISCRETION	L	L	N
169		[free text]	N	L	R
170		[free text]	D	H	R
194		[free text]	N	L	Y
178		[free text]	N	L	N
195		[free text]	L	L	R
196		[free text]	N	M	W/U
197		[free text]	U	M	W/U
198		[free text]	D	H	W/U
199		[free text]	N	M	W/U
200		[free text]	L	L	R
201		[free text]	N	M	W/U
202		[free text]	D	H	W/U
203		[free text]	N	M	R
204		[free text]	N	M	Y
183		[free text]	N	M	N
205		[free text]	N	M	A/N
206		[free text]	L	N	Y

187		[free text]	L	N	N
207		[free text]	L	L	Y
208		[free text]	L	L	N

Table 7-12: Additional Messages (uplink)

7.7.4 Downlink Messages

	Message Intent/Use	Message Element	URG	ALRT	RESP
0	The instruction 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 7-13: Responses (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Request to fly at the specified altitude.	REQUEST [altitude]	N	L	Y
7	Request to fly at an altitude within the specified altitude interval.	REQUEST BLOCK [altitude] TO [altitude]	N	L	Y
8	Request to cruise climb to the specified altitude.	REQUEST CRUISE CLIMB TO [altitude]	N	L	Y
9	Request to climb to the specified altitude.	REQUEST CLIMB TO [altitude]	N	L	Y
10	Request to descend to the specified altitude.	REQUEST DESCENT TO [altitude]	N	L	Y
11	Request that at the specified position a climb to the specified altitude be approved.	AT [position] REQUEST CLIMB TO [altitude]	N	L	Y
12	Request that at the specified position a descent to the specified altitude be approved.	AT [position] REQUEST DESCENT TO [altitude]	N	L	Y
13	Request that at the specified time a climb to the specified altitude be approved.	AT [time] REQUEST CLIMB TO [altitude]	N	L	Y
14	Request that at the specified time a descent to the specified altitude be approved.	AT [time] REQUEST DESCENT TO [altitude]	N	L	Y
69	Request that a descent be approved on a see-and-avoid basis.	REQUEST VMC DESCENT	N	L	Y

Table 7-14: Vertical 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 [distance offset] [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 [distance offset] [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 [distance offset] [direction] OF ROUTE	N	L	Y

Table 7-15: Lateral Off-Set 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 7-16: Speed 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 7-17: Voice Contact 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 [distance offset] [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 7-18: Route Modification Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
28	Notification of leaving the specified altitude.	LEAVING [altitude]	N	L	N
29	Notification of climbing to the specified altitude.	CLIMBING TO [altitude]	N	L	N
30	Notification of descending to the specified altitude.	DESCENDING TO [altitude]	N	L	N
31	Notification of passing the specified position.	PASSING [position]	N	L	N
78	At the specified time, the aircraft's position was as specified.	AT [time] [distance] [to/from] [position]	N	L	N
32	Notification of the present altitude.	PRESENT ALTITUDE[altitude]	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
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 altitude.	ALTITUDE [altitude]	N	L	N
72	Notification that the aircraft has reached the specified altitude.	REACHING [altitude]	N	L	N
76	Notification that the aircraft has reached an altitude within the specified altitude range.	REACHING BLOCK [altitude] TO [altitude]	N	L	N
38	Read-back of the assigned altitude.	ASSIGNED ALTITUDE[altitude]	N	M	N
77	Read-back of the assigned altitude range.	ASSIGNED BLOCK [altitude] TO [altitude]	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

46	Clarification of time over previously reported waypoint.	REPORTED WAYPOINT [time]	N	L	N
47	The specified beacon code has been selected and the SSR transponder is operational.	SQUAWKING [beacon code]	N	L	N
48	Position report.	POSITION REPORT [position report]	N	M	N
79	The code of the latest ATIS received is as specified.	ATIS [atis code]	N	L	N
89	The specified ICAO unit is being monitored on the specified frequency.	MONITORING [icaounitname] [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 altitude~~level.~~	PREFERRED ALTITUDE ~~LEVEL~~ [altitude]	L	L	N
109	Notification of the preferred time to commence descent for approach	TOP OF DESCENT [time]	L	L	N
110	Notification of the preferred position to commence descent for approach	TOP OF DESCENT [position]	L	L	N
111	Notification of the preferred time and position to commence descent for approach	TOP OF DESCENT [time] position]	L	L	N

Table 7-19: Reports (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
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 ALTITUDE	L	L	Y
53	Request for the earliest time at which a clearance to climb can be expected.	WHEN CAN WE EXPECT HIGHER ALTITUDE	L	L	Y
54	Request for the earliest time at which a clearance to cruise climb to the specified altitude can be expected.	WHEN CAN WE EXPECT CRUISE CLIMB TO [altitude]	L	L	Y
87	Request for the earliest time at which a clearance to climb to the specified altitude can be expected.	WHEN CAN WE EXPECT CLIMB TO [altitude]	L	L	Y
88	Request for the earliest time at which a clearance to descend to the specified altitude can be expected.	WHEN CAN WE EXPECT DESCENT TO [altitude]	L	L	Y

Table 7-20: Negotiation Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
55	Urgency prefix.	PAN PAN PAN	U	H	N
56	Distress prefix.	MAYDAY MAYDAY MAYDAY	D	H	N
112	Indicates specifically that the aircraft is being subjected to unlawful interference.	SQUAWKING 7500	U	H	N
57	Notification of fuel remaining and number of persons on board.	[remaining fuel] OF FUEL REMAINING AND [persons~~souls~~ on board] PERSONS~~SOULS~~ ON BOARD	U	H	N
58	Notification that the pilot wishes to cancel the emergency condition.	CANCEL EMERGENCY	U	M	N
59	Notification that the aircraft is diverting to the specified position via the specified route.	DIVERTING TO [position] VIA [route clearance]	U	H	N
60	Notification that the aircraft is deviating the specified distance in the specified direction off the cleared route and maintaining a parallel track.~~Notification that the aircraft is diverting the specified distance in the specified direction off the cleared route.~~	OFFSETTING [distance offset] [direction] OF ROUTE	U	H	N
61	Notification that the aircraft is descending to the specified altitude.	DESCENDING TO [altitude]	U	H	N
80	Notification that the aircraft is deviating from the cleared route by the specified distance in the specified direction.	DEVIATING [distance offset] [direction] OFF ROUTE	U	H	N

Table 7-21: Emergency 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~~to a~~ ground facility that is not the Current Data Authority.~~.~~	NOT CURRENT DATA AUTHORITY	L	L	N
99	A system generated message to inform a ground facility that it is now the Current Data Authority	CURRENT DATA AUTHORITY	L	L	N
107	A system generated message sent to a ground system that tries to connect to an aircraft when a current data authority has not designated the ground system as the NDA.	NOT AUTHORIZED NEXT DATA AUTHORITY	L	L	N
64	Notification to the ground system that the specified ATSU is the current data authority.	[icao facility designator]	L	L	N
73	A system generated message indicating the software version number.	[version number]	L	L	N
100	Notification to the ground system that the aircraft system has received the message to which the logical acknowledgment refers.	LOGICAL ACKNOWLEDGMENT	N	M	N

Table 7-22: System Management Messages (downlink)

	Message Intent/Use	**Message Element**	**URG**	**ALRT**	**RESP**
65	Used to explain reasons for aircraft operator's message.	DUE TO WEATHER	L	L	N
66	Used to explain reasons for aircraft operator's message.	DUE TO AIRCRAFT PERFORMANCE	L	L	N
74	States a desire by the aircraft operator to provide his/her own separation under see and avoid conditions.	REQUEST TO MAINTAIN OWN SEPARATION AND VMC	L	L	Y
75	Used in conjunction with another message to indicate that the operator wishes to execute request when the air crew is prepared to do so.	AT PILOTS DISCRETION	L	L	N
101	Allows the aircraft operator to indicate a desire for termination of CPDLC service with the current data authority.	REQUEST END OF SERVICE	L	L	Y
103	Allows the aircraft operator to indicate that he has canceled IFR flight plan.	CANCELLING IFR	N	L	Y
108	Notification that de-icing action has been completed.	DE-ICING COMPLETE	L	L	N
67		[free text]	N	L	N
68		[free text]	D	H	Y
90		[free text]	N	M	N
91		[free text]	N	L	Y
92		[free text]	L	L	Y
93		[free text]	U	H	N
94		[free text]	D	H	N
95		[free text]	U	M	N
96		[free text]	U	L	N
97		[free text]	L	L	N
98		[free text]	N	N	N

Table 7-23: Additional Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
81	We can accept the specified altitude at the specified time.	WE CAN ACCEPT [altitude] AT [time]	L	L	N
82	We cannot accept the specified altitude.	WE CANNOT ACCEPT [altitude]	L	L	N
83	We can accept the specified speed at the specified time.	WE CAN ACCEPT [speed] AT [time]	L	L	N
84	We cannot accept the specified speed.	WE CANNOT ACCEPT [speed]	L	L	N
85	We can accept a parallel track offset the specified distance in the specified direction at the specified time.	WE CAN ACCEPT [direction] [distance offset] at [time]	L	L	N
86	We cannot accept a parallel track offset the specified distance in the specified direction.	WE CANNOT ACCEPT [direction] [distance offset]	L	L	N

Table 7-24: Negotiation Responses (downlink)

7.8 Message Variables Range and Resolution

7.8.1 A CPDLC-user shall interpret CPDLC message element variables as defined in Chapter 4.~~Table 7-25~~.

~~*Table 7-25: CPDLC Message Element Range and Resolution*~~