APPENDIX E: CORRECTIONS TO WORD VERSION 3.0 OF THE CONTROLLER PILOT DATA LINK COMMUNICATION (CPDLC) SARPS

2.3.1. INTRODUCTION

2.3.1.1 Introduction

2.3.1.1.1 This application allows data link communication between controllers and pilots.

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

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

Note 1. — Structure:

a) 2.3 1: INTRODUCTION contains the ~~document's purpose and~~ structure of 2.3, and a summary of the functional capabilities of CPDLC.

b) 2.3..2: GENERAL REQUIREMENTS contains the CPDLC version number, and error processing requirements.

c) 2.3. 3: ABSTRACT SERVICE DEFINITION contains the description of the abstract service provided by the CPDLC Application Service Element (CPDLC-ASE).

d) 2.3. 4: FORMAL DEFINITION OF MESSAGES contains the formal definition of messages exchanged by CPDLC ASEs using Abstract Syntax Notation Number One (ASN.1).

e) 2.3. 5: PROTOCOL DEFINITION describes the exchanges of messages allowed by the CPDLC protocol, as well as time constraints and CPDLC-ASE protocol descriptions and state tables.

f) 2.3. 6: COMMUNICATION REQUIREMENTS contains the requirements that the CPDLC application imposes on the underlying communication system.

g) 2.3. 7: CPDLC USER REQUIREMENTS contains requirements imposed on the user of the CPDLC ASE service and message description tables.

h) 2.3. 8: SUBSETTING RULES defines the conformant subsets for the CPDLC-ASE.

Note 2.~~:~~ — Functional Descriptions

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

- *1) general information exchange;*
- *2) clearance*
  - *i) delivery,*
  - *ii) request, and*
  - *iii) response;*
- *3) altitude/identity surveillance;*
- *4) monitoring of current/planned position;*
- *5) advisories*
  - *i) request and*
  - *ii) delivery;*

6) system management functions; and

7) emergency situations.

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

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

*d) The **Ground Forward Function** provides the capability for a ground system to forward information received in a CPDLC message to another ground system. The ground forwarding function can be used by the controlling data authority to forward an aircraft request to the next data authority, so that an aircraft does not need to issue the same request again. This function can also be used by a downstream data authority to pass a message to a current data authority for transmission by the current data authority to an aircraft. This information is exchanged using CPDLC message(s). It is a one-way forwarding of information with an indication of success, failure or non-support from the receiving ground system.*

Note 3. — See 2.3.7 for detailed CPDLC message intent/use descriptions.

2.3.2. GENERAL REQUIREMENTS

2.3.2.1 CPDLC~~CPLDC~~ ASE Version Number

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

2.3.2.2 Error Processing Requirements

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

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

2.3.3. THE ABSTRACT SERVICE

2.3.3.1 Service Description

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

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

~~Implementations of the CPDLC-ASE shall exhibit the behavior defined in 2.3.3.2 to 2.3.3.12.~~

Note 2~~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 3~~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 2.3.7.

Note 4~~3~~. — Figure 2.3.~~3~~ 3-1 shows the functional model of the CPDLC Application. The functional modules identified in this model are the following :

a) the CPDLC-user,
b) the CPDLC Application Entity (CPDLC-AE) service ~~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~~.

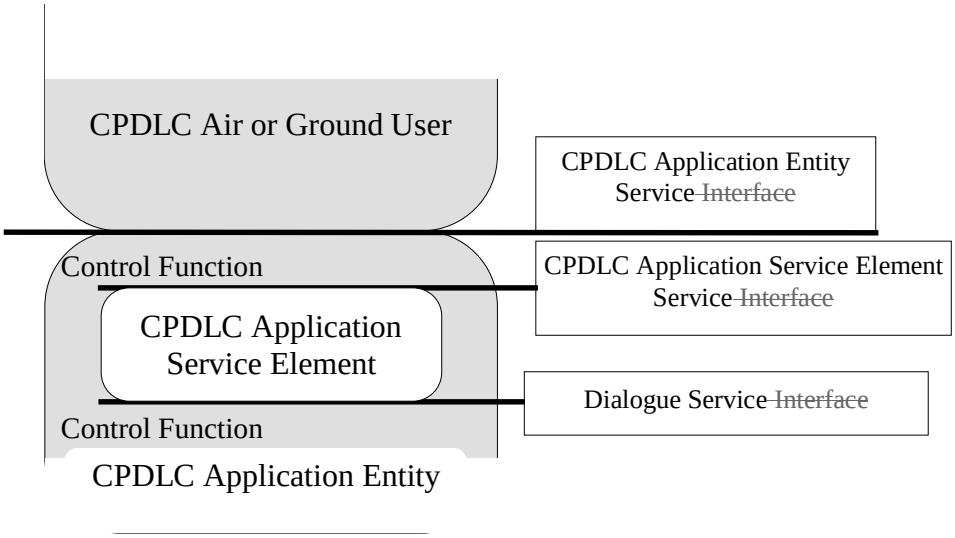

Figure 2.3.3-1 Functional Model of the CPDLC Application

Note 4. — 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 possible through the ~~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 5. — 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 6. — 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 7. — ~~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 service primitives of other abstract elements. ~~other elements which interface with it. The part of the CPDLC-CF which is relevant from the point of view of the CPDLC application, i.e. the part between the CPDLC-user and the CPDLC-ASE, will map CPDLC-AE service primitives to CPDLC-ASE service primitives transparently~~.

*Note 8. — The CPDLC-ASE has two abstract boundaries with the CPDLC-CF: the CPDLC-ASE service and the dialogue service. The CPDLC-CF maps CPDLC-AE service primitives to CPDLC-ASE service primitives transparently and vice versa. The CPDLC-CF maps dialogue service primitives to other abstract elements in the CPDLC-AE and the underlying communication service, and vice versa.~~*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].*

2.3.3.2 The CPDLC-ASE Abstract Service

~~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.~~

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

a) *CPDLC-start service* as defined in 2.3.3.5,
b) *DSC-start service* as defined in 2.3.3.6,
c) *CPDLC-message service* as defined in 2.3.3.7,
d) *CPDLC-end service* as defined in 2.3.3.8,
e) *DSC-end service* as defined in 2.3.3.9,
f) *CPDLC-forward service* as defined in 2.3.3.10,
g) *CPDLC-user-abort service* as defined in 2.3.3.11, and
h) *CPDLC-provider-abort service* as defined in 2.3.3.12.

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

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

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

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

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

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

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

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

2.3.3.3 CPDLC-start Service

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

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

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

Table 2.3.3-1. CPDLC-start Service Parameters

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

2.3.3.3.2 Called Peer Identifier

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

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

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

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

2.3.3.3.3 Calling Peer Identifier

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

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

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

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

2.3.3.3.4 CPDLC Message

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

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

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

2.3.3.3.5 Reject Reason

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

2.3.3.3.5.1 If, 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.

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

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

2.3.3.3.6 Result

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

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

2.3.3.3.7 Class of Communication Service

Note. — 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.

2.3.3.3.7.1 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".

Note. — Class of Communication service parameter values are detailed in 1.2.

2.3.3.4 DSC-start Service

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

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

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

Table 2.3.3-2. DSC-start Service Parameters

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	M		

2.3.3.4.2 ICAO Facility Designator

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

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

2.3.3.4.3 Aircraft Identifier

2.3.3.4.3.1 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.

2.3.3.4.4 CPDLC Message

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

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

2.3.3.4.5 Reject Reason

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

2.3.3.4.5.1 If~~,~~ 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.

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

2.3.3.4.6 Result

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

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

2.3.3.4.7 Class of Communication Service

Note. — 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.

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

Note. — Class of Communication service parameter values are detailed in 1.2.

2.3.3.5 CPDLC-message Service

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

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

Table 2.3.3-3. CPDLC-message Service Parameters

Parameter Name	Req	Ind
CPDLC Message	M	M(=)

2.3.3.5.2 CPDLC Message

Note. — This parameter contains a CPDLC message.

2.3.3.5.2.1 If the CPDLC-message service is invoked by the CPDLC-ground-user, t~~T~~he *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCUplinkMessage~~, if provided by the CPDLC-ground-user~~.

2.3.3.5.2.2 If the CPDLC-message service is invoked by the CPDLC-air-user, t~~T~~he *CPDLC Message* parameter value shall conform to the ASN.1 abstract syntax ATCDownlinkMessage~~, if provided by the CPDLC-air-user~~.

2.3.3.6 CPDLC-end Service

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

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

Table 2.3.3-4. CPDLC-end Service Parameters

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

2.3.3.6.2 CPDLC Message

Note. — This parameter contains a CPDLC message.

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

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

2.3.3.6.3 Result

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

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

2.3.3.7 DSC-end Service

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

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

Table 2.3.3-5. DSC-end Service Parameters

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

2.3.3.7.2 CPDLC Message

Note. — This parameter contains a CPDLC message.

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

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

2.3.3.7.3 Result

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

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

2.3.3.8 CPDLC-forward Service

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

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

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

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		
Result			M

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

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

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

2.3.3.8.3 Called ICAO Facility Designator

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

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

2.3.3.8.4 Calling ICAO Facility Designator

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

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

2.3.3.8.5 ASE Version Number

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

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

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

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

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

2.3.3.8.6 CPDLC Message

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

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

2.3.3.8.7 Class of Communication Service

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

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

Note. — Class of Communication service parameter values are detailed in 1.2~~2.1~~.

2.3.3.8.8 Result

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

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

2.3.3.9 CPDLC-user-abort Service

Note. — 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.

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

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

Parameter Name	Req	Ind
Reason	U	M

2.3.3.9.2 Reason

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

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

2.3.3.10 CPDLC-provider-abort Service

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

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

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

Parameter Name	Ind
Reason	M

2.3.3.10.2 Reason

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

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

2.3.4. FORMAL DEFINITIONS OF MESSAGES

2.3.4.1 Encoding/Decoding Rules

2.3.4.1.1 A CPDLC-air-ASE shall be capable of encoding AircraftPDUs APDUs and decoding GroundPDUs APDUs.

2.3.4.1.2 A CPDLC-ground-ASE shall be capable of encoding and decoding GroundPDUs APDUs and decoding AircraftPDUs APDUs.

2.3.4.2 CPDLC ASN.1 Abstract Syntax

2.3.4.2.1 The abstract syntax of the CPDLC protocol data units shall comply with the description contained in the ASN.1 module CPDLCMessageSetVersion1 ~~(~~conforming to ISO/IEC 8824~~[1])~~, as defined in this section.

```
CPDLCMessageSetVersion1 DEFINITIONS::=

BEGIN

-- ------------------------------------------------------------------------------
-- Ground Generated Messages - Top level
-- ------------------------------------------------------------------------------
GroundPDUs ::= CHOICE
        {
        abortUser           [0]     CPDLCUserAbortReason,
        abortProvider       [1]     CPDLCProviderAbortReason,
        startup             [21]    UplinkMessage,
        send                [32]    ATCUplinkMessage,
        forward             [43]    ATCForwardMessage,
        forwardresponse     [54]    ATCForwardResponse,
        ...
        }

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

ATCUplinkMessage ::= SEQUENCE
        {
        header              ATCMessageHeader,
        elementIds          SEQUENCE SIZE (1..5) OF ATCUplinkMsgElementId
        }

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

ForwardHeader ::= SEQUENCE
```

```
        {
        dateTime                DateTimeGroup,
        aircraftID              AircraftFlightIdentification,
        aircraftdAdressairframeID       AircraftAddressAirFrameID
        }

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

ATCForwardResponse ::= ENUMERATED
        {
        success                 (0),
        service-not-supported   (1),
        version-not-equal       (2),.
        ...
        }

-- ------------------------------------------------------------------------------
-- Aircraft Generated Messages - Top level
-- ------------------------------------------------------------------------------
AircraftPDUs::= CHOICE
        {
        abortUser       ———[0]  CPDLCUserAbortReason,
        abortProvider   [1]     CPDLCProviderAbortReason,
        startdown       [21]    StartDownMessage,
        send            [32]    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,
```

```
        elementIds              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~~
~~        {~~
~~        cPDLC-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),~~
~~        commanded-termination                           (11),~~
~~        ...~~
~~        }~~
CPDLCUserAbortReason ::= ENUMERATED
        {
        undefined~~cPDLC-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),

        commanded-termination                           (5),
        ...
        }

CPDLCProviderAbortReason ::= ENUMERATED
        {
        timer-expired                                   (0),
        undefined-error                                 (1),
        invalid-PDU                                     (2),
        not-permitted-PDU                               (3),
```

```
        communication-service-error          (4),
        communication-service-failure        (5),
        ...
        }

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 [levelaltitude]                          Urg(L)/Alr(L)/Resp( R )
        uM6LevelAltitude                                [6] LevelAltitude,

--      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] uM12Position	Urg(L)/Alr(L)/Resp(R) [12] Position,
--	AT [time] EXPECT CLIMB TO [level~~altitude~~] uM13TimeLevel~~Altitude~~	Urg(L)/Alr(L)/Resp(R) [13] TimeLevel~~Altitude~~,
--	AT [position] EXPECT CLIMB TO [level~~altitude~~] uM14PositionLevel~~Altitude~~	Urg(L)/Alr(L)/Resp(R) [14] PositionLevel~~Altitude~~,
--	AT [time] EXPECT DESCENT TO [level~~altitude~~] uM15TimeLevel~~Altitude~~	Urg(L)/Alr(L)/Resp(R) [15] TimeLevel~~Altitude~~,
--	AT [position] EXPECT DESCENT TO [level~~altitude~~] uM16PositionLevel~~Altitude~~	Urg(L)/Alr(L)/Resp(R) [16] PositionLevel~~Altitude~~,
--	AT [time] EXPECT CRUISE CLIMB TO [level~~altitude~~] uM17TimeLevel~~Altitude~~	Urg(L)/Alr(L)/Resp(R) [17] TimeLevel~~Altitude~~,
--	AT [position] EXPECT CRUISE CLIMB TO [level~~altitude~~] uM18PositionLevel~~Altitude~~	Urg(L)/Alr(L)/Resp(R) [18] PositionLevel~~Altitude~~,
--	MAINTAIN [level~~altitude~~] uM19Level~~Altitude~~	Urg(N)/Alr(M)/Resp(W/U) [19] Level~~Altitude~~,
--	CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U) uM20Level~~Altitude~~	[20] Level~~Altitude~~,
--	AT [time] CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U) uM21TimeLevel~~Altitude~~	[21] TimeLevel~~Altitude~~,
--	AT [position] CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U) uM22PositionLevel~~Altitude~~	[22] PositionLevel~~Altitude~~,
--	DESCEND TO ~~AND MAINTAIN~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U) uM23Level~~Altitude~~	[23] Level~~Altitude~~,
--	AT [time] DESCEND TO ~~AND MAINTAIN~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U) uM24TimeLevel~~Altitude~~	[24] TimeLevel~~Altitude~~,
--	AT [position] DESCEND TO ~~AND MAINTAIN~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U) uM25PositionLevel~~Altitude~~	[25] PositionLevel~~Altitude~~,
--	CLIMB TO REACH [level~~altitude~~] BY [time] uM26Level~~Altitude~~Time	Urg(N)/Alr(M)/Resp(W/U) [26] Level~~Altitude~~Time,
--	CLIMB TO REACH [level~~altitude~~] BY [position] uM27Level~~Altitude~~Position	Urg(N)/Alr(M)/Resp(W/U) [27] Level~~Altitude~~Position,

-- DESCEND TO REACH [level~~altitude~~] BY [time] Urg(N)/Alr(M)/Resp(W/U)
uM28Level~~Altitude~~Time [28] Level~~Altitude~~Time,

-- DESCEND TO REACH [level~~altitude~~] BY [position] Urg(N)/Alr(M)/Resp(W/U)
uM29Level~~Altitude~~Position [29] Level~~Altitude~~Position,

-- MAINTAIN BLOCK [level~~altitude~~] TO [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U)
uM30Level~~Altitude~~Level~~Altitude~~ [30] Level~~Altitude~~Level~~Altitude~~,

-- CLIMB TO AND MAINTAIN BLOCK [level~~altitude~~] TO [level~~altitude~~]
Urg(N)/Alr(M)/Resp(W/U)
uM31Level~~Altitude~~Level~~Altitude~~ [31] Level~~Altitude~~Level~~Altitude~~,

-- DESCEND TO AND MAINTAIN BLOCK [level~~altitude~~] TO [level~~altitude~~]
Urg(N)/Alr(M)/Resp(W/U)
uM32Level~~Altitude~~Level~~Altitude~~ [32] Level~~Altitude~~Level~~Altitude~~,

-- *Reserved*~~CRUISE [altitude]~~ ~~Urg(N)/Alr(M)/Resp(W/U)~~
uM33null~~Altitude~~ [33] NULL~~Altitude~~,

-- CRUISE CLIMB TO [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U)
uM34Level~~Altitude~~ [34] Level~~Altitude~~,

-- CRUISE CLIMB ABOVE [level~~altitude~~] Urg(N)/Alr(M)/Resp(W/U)
uM35Level~~Altitude~~ [35] Level~~Altitude~~,

-- EXPEDITE CLIMB TO [level~~altitude~~] Urg(U)/Alr(M)/Resp(W/U)
uM36Level~~Altitude~~ [36] Level~~Altitude~~,

-- EXPEDITE DESCENT TO [level~~altitude~~] Urg(U)/Alr(M)/Resp(W/U)
uM37Level~~Altitude~~ [37] Level~~Altitude~~,

-- IMMEDIATELY CLIMB TO [level~~altitude~~] Urg(D)/Alr(H)/Resp(W/U)
uM38Level~~Altitude~~ [38] Level~~Altitude~~,

-- IMMEDIATELY DESCEND TO [level~~altitude~~] Urg(D)/Alr(H)/Resp(W/U)
uM39Level~~Altitude~~ [39] Level~~Altitude~~,

-- IMMEDIATELY STOP CLIMB AT [level~~altitude~~] Urg(D)/Alr(H)/Resp(W/U)
uM40Level~~Altitude~~ [40] Level~~Altitude~~,

-- IMMEDIATELY STOP DESCENT AT [level~~altitude~~] Urg(D)/Alr(H)/Resp(W/U)
uM41Level~~Altitude~~ [41] Level~~Altitude~~,

-- EXPECT TO CROSS [position] AT [level~~altitude~~] Urg(L)/Alr(L)/Resp(R)
uM42PositionLevel~~Altitude~~ [42] PositionLevel~~Altitude~~,

-- EXPECT TO CROSS [position] AT OR ABOVE [level~~altitude~~]
Urg(L)/Alr(L)/Resp(R)
uM43PositionLevel~~Altitude~~ [43] PositionLevel~~Altitude~~,

-- EXPECT TO CROSS [position] AT OR BELOW [level~~altitude~~]
Urg(L)/Alr(L)/Resp(R)

uM44PositionLevelAltitude [44] PositionLevelAltitude,

-- EXPECT TO CROSS [position] AT AND MAINTAIN [levelaltitude]
Urg(L)/Alr(L)/Resp(R)
uM45PositionLevelAltitude [45] PositionLevelAltitude,

-- CROSS [position] AT [levelaltitude] Urg(N)/Alr(M)/Resp(W/U)
uM46PositionLevelAltitude [46] PositionLevelAltitude,

-- CROSS [position] AT OR ABOVE [levelaltitude] Urg(N)/Alr(M)/Resp(W/U)
uM47PositionLevelAltitude [47] PositionLevelAltitude,

-- CROSS [position] AT OR BELOW [levelaltitude] Urg(N)/Alr(M)/Resp(W/U)
uM48PositionLevelAltitude [48]PositionLevelAltitude,

-- CROSS [position] AT AND MAINTAIN [levelaltitude] Urg(N)/Alr(M)/Resp(W/U)
uM49PositionLevelAltitude [49] PositionLevelAltitude,

-- CROSS [position] BETWEEN [levelaltitude] AND [levelaltitude]
Urg(N)/Alr(M)/Resp(W/U)
uM50PositionLevelAltitudeLevelAltitude [50]
PositionLevelAltitudeLevelAltitude,

-- 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 [levelaltitude] Urg(N)/Alr(M)/Resp(W/U)
uM58PositionTimeLevelAltitude [58] PositionTimeLevelAltitude,

-- CROSS [position] AT OR BEFORE [time] AT [levelaltitude]
Urg(N)/Alr(M)/Resp(W/U)
uM59PositionTimeLevelAltitude [59] PositionTimeLevelAltitude,

-- CROSS [position] AT OR AFTER [time] AT [levelaltitude]
Urg(N)/Alr(M)/Resp(W/U)

uM60PositionTimeLevel~~Altitude~~ _____[60] PositionTimeLevel~~Altitude~~,

-- CROSS [position] AT AND MAINTAIN [level~~altitude~~] AT [speed]
_____Urg(N)/Alr(M)/Resp(W/U)
uM61PositionLevel~~Altitude~~Speed _____[61] PositionLevel~~Altitude~~Speed,

-- AT [time] CROSS [position] AT AND MAINTAIN [level~~altitude~~]
Urg(N)/Alr(M)/Resp(W/U)
uM62TimePositionLevel~~Altitude~~ _____[62] TimePositionLevel~~Altitude~~,

-- AT [time] CROSS [position] AT AND MAINTAIN [level~~altitude~~] AT [speed]
-- Urg(N)/Alr(M)/Resp(W/U)
uM63TimePositionLevel~~Altitude~~Speed [63] TimePositionLevel~~Altitude~~Speed,

-- 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 [levelaltitude] PROCEED DIRECT TO [position] Urg(N)/Alr(M)/Resp(W/U)
uM78LevelAltitudePosition [78] LevelAltitudePosition,

-- 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 [levelaltitude] EXPECT DIRECT TO [position] Urg(L)/Alr(L)/Resp(R)
uM90LevelAltitudePosition [90] LevelAltitudePosition,

-- HOLD AT [position] MAINTAIN [levelaltitude] INBOUND TRACK [degrees][direction]
-- TURNS [legtype] Urg(N)/Alr(M)/Resp(W/U)
uM91HoldClearance [91] HoldClearance,

-- HOLD AT [position] AS PUBLISHED MAINTAIN [levelaltitude]
Urg(N)/Alr(M)/Resp(W/U)
uM92PositionLevelAltitude [92] PositionLevelAltitude,

-- EXPECT FURTHER CLEARANCE AT [time] Urg(L)/Alr(L)/Resp(R)

	uM93Time	[93] Time,
--	TURN [direction] HEADING [degrees] uM94DirectionDegrees	Urg(N)/Alr(M)/Resp(W/U) [94] DirectionDegrees,
--	TURN [direction] GROUND TRACK [degrees] uM95DirectionDegrees	Urg(N)/Alr(M)/Resp(W/U) [95] DirectionDegrees,
--	FLY PRESENT HEADING uM96NULL	Urg(N)/Alr(M)/Resp(W/U) [96] NULL,
--	AT [position] FLY HEADING [degrees] uM97PositionDegrees	Urg(N)/Alr(M)/Resp(W/U) [97] PositionDegrees,
--	IMMEDIATELY TURN [direction] HEADING [degrees] uM98DirectionDegrees	Urg(D)/Alr(H)/Resp(W/U) [98] DirectionDegrees,
--	EXPECT [procedureName] uM99ProcedureName	Urg(L)/Alr(L)/Resp(R) [99] ProcedureName,
--	AT [time] EXPECT [speed] uM100TimeSpeed	Urg(L)/Alr(L)/Resp(R) [100] TimeSpeed,
--	AT [position] EXPECT [speed] uM101PositionSpeed	Urg(L)/Alr(L)/Resp(R) [101] PositionSpeed,
--	AT [level~~altitude~~] EXPECT [speed] uM102Level~~Altitude~~Speed	Urg(L)/Alr(L)/Resp(R) [102] Level~~Altitude~~Speed,
--	AT [time] EXPECT [speed] TO [speed] uM103TimeSpeedSpeed	Urg(L)/Alr(L)/Resp(R) [103] TimeSpeedSpeed,
--	AT [position] EXPECT [speed] TO [speed] uM104PositionSpeedSpeed	Urg(L)/Alr(L)/Resp(R) [104] PositionSpeedSpeed,
--	AT [level~~altitude~~] EXPECT [speed] TO [speed] uM105Level~~Altitude~~SpeedSpeed	Urg(L)/Alr(L)/Resp(R) [105] Level~~Altitude~~SpeedSpeed,
--	MAINTAIN [speed] uM106Speed	Urg(N)/Alr(M)/Resp(W/U) [106] Speed,
--	MAINTAIN PRESENT SPEED uM107NULL	Urg(N)/Alr(M)/Resp(W/U) [107] NULL,
--	MAINTAIN [speed] OR GREATER uM108Speed	Urg(N)/Alr(M)/Resp(W/U) [108] Speed,
--	MAINTAIN [speed] OR LESS uM109Speed	Urg(N)/Alr(M)/Resp(W/U) [109] Speed,
--	MAINTAIN [speed] TO [speed] uM110SpeedSpeed	Urg(N)/Alr(M)/Resp(W/U) [110] SpeedSpeed,

--	INCREASE SPEED TO [speed] uM111Speed	Urg(N)/Alr(M)/Resp(W/U) [111] Speed,
--	INCREASE SPEED TO [speed] OR GREATER uM112Speed	Urg(N)/Alr(M)/Resp(W/U) [112] Speed,
--	REDUCE SPEED TO [speed] uM113Speed	Urg(N)/Alr(M)/Resp(W/U) [113] Speed,
--	REDUCE SPEED TO [speed] OR LESS uM114Speed	Urg(N)/Alr(M)/Resp(W/U) [114] Speed,
--	DO NOT EXCEED [speed] uM115Speed	Urg(N)/Alr(M)/Resp(W/U) [115] Speed,
--	RESUME NORMAL SPEED uM116NULL	Urg(N)/Alr(M)/Resp(W/U) [116] NULL,
--	CONTACT [~~icao~~unitname] [frequency] uM117~~ICAO~~UnitNameFrequency	_____Urg(N)/Alr(M)/Resp(W/U) [117] ~~ICAO~~UnitNameFrequency,
--	AT [position] CONTACT [~~icao~~unitname] [frequency] uM118Position~~ICAO~~UnitNameFrequency	_____Urg(N)/Alr(M)/Resp(W/U) [118] Position~~ICAO~~UnitNameFrequency,
--	AT [time] CONTACT [~~icao~~unitname] [frequency] uM119Time~~ICAO~~UnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) _____[119] Time~~ICAO~~UnitNameFrequency,
--	MONITOR [~~icao~~unitname] [frequency] uM120~~ICAO~~UnitNameFrequency	_____Urg(N)/Alr(M)/Resp(W/U) [120] ~~ICAO~~UnitNameFrequency,
--	AT [position] MONITOR [~~icao~~unitname] [frequency] uM121Position~~ICAO~~UnitNameFrequency	_____Urg(N)/Alr(M)/Resp(W/U) [121] Position~~ICAO~~UnitNameFrequency,
--	AT [time] MONITOR [~~icao~~unitname] [frequency] uM122Time~~ICAO~~UnitNameFrequency	Urg(N)/Alr(M)/Resp(W/U) _____[122] Time~~ICAO~~UnitNameFrequency,
--	SQUAWK [~~beacon~~code] uM123~~Beacon~~Code	Urg(N)/Alr(M)/Resp(W/U) _____[123] ~~Beacon~~Code,
--	STOP SQUAWK uM124NULL	Urg(N)/Alr(M)/Resp(W/U) [124] NULL,
--	SQUAWK MODE CHARLIE uM125NULL	Urg(N)/Alr(M)/Resp(W/U) [125] NULL,
--	STOP SQUAWK MODE CHARLIE uM126NULL	Urg(N)/Alr(M)/Resp(W/U) [126] NULL,
--	REPORT BACK ON ROUTE uM127NULL	Urg(N)/Alr(M)/Resp(W/U~~R~~) [127] NULL,

-- REPORT LEAVING [levelaltitude] Urg(N)/Alr(M)/Resp(-W/UR)
uM128LevelAltitude [128] LevelAltitude,

-- REPORT MAINTAININGWHEN LEVEL AT [levelaltitude]
Urg(N)/Alr(M)/Resp(W/UR)
uM129LevelAltitude [129] LevelAltitude,

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

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

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

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

-- REPORT [speedtype] [speedtype] [speedtype]SPEED
Urg(N)/Alr(M)/Resp(Y)
uM134SpeedTypeSpeedTypeSpeedType [134] SpeedTypeSpeedTypeSpeedType,

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

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

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

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

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

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

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

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

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

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

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

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

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

--      WHEN CAN YOU ACCEPT [levelaltitude]             Urg(N)/Alr(M)/Resp(Y)
        uM148LevelAltitude                                  [148] LevelAltitude,

--      CAN YOU ACCEPT [levelaltitude] AT [position]    Urg(N)/Alr(M)/Resp(A/N)
        uM149LevelAltitudePosition                          [149] LevelAltitudePosition,

--      CAN YOU ACCEPT [levelaltitude] AT [time]            Urg(N)/Alr(M)/Resp(A/N)
        uM150LevelAltitudeTime                              [150] LevelAltitudeTime,

--      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] ATISCode,

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

--      NEXT DATA AUTHORITY [icaofacilitydesignationdesignator]  Urg(L)/Alr(N)/Resp(N)
        uM160ICAOFacilityDesignationDesignator                   [160]
ICAOFacilityDesignationDesignator,

--      END SERVICE                                     Urg(L)/Alr(N)/Resp(N)
        uM161NULL                                       [161] NULL,

--      SERVICE UNAVAILABLE                             Urg(L)/Alr(L)/Resp(N )
```

uM162NULL [162] NULL,

-- [~~icao~~facility<u>designation</u>~~designator~~] Urg(L)/Alr(N)/Resp(N)
uM163~~ICAO~~Facility<u>Designation</u>~~Designator~~ _____[163]
~~ICAO~~Facility<u>Designation</u>~~Designator~~,

-- 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 [<u>level</u>~~altitude~~] Urg(N)/Alr(M)/Resp(<u>W/U</u>~~R~~)
uM175<u>Level</u>~~Altitude~~ [175] <u>Level</u>~~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 [levelaltitude] TO [levelaltitude]
                                                    Urg(N)/Alr(M)/Resp(W/UR)
        uM180LevelAltitudeLevelAltitude             [180] LevelAltitudeLevelAltitude,

--      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 [levelaltitude]
                                                    Urg(N)/Alr(M)/Resp(W/U)
        uM185PositionLevelAltitude                  [185] PositionLevelAltitude,

--      AFTER PASSING [position] DESCEND TO AND MAINTAIN [levelaltitude]
                                                    Urg(N)/Alr(M)/Resp(W/U)
        uM186PositionLevelAltitude                  [186] PositionLevelAltitude,

--      [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 [levelaltitude] BY [time]             Urg(N)/Alr(M)/Resp(W/U)
        uM192LevelAltitudeTime                      [192] LevelAltitudeTime,

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

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

--      [freetext]                                  Urg(L)/Alr(L)/Resp(R)
        uM195FreeText                               [195] FreeText,

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

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

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

-- [freetext] Urg(N)/Alr(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 [levelaltitude] BY [position] Urg(N)/Alr(M)/Resp(W/U)
uM209LevelAltitudePosition [209] LevelAltitudePosition,

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

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

-- [icaofacilitydesignationdesignator] ATIS [atiscode] CURRENT Urg(N)/Alr(M)/Resp(R)
uM212ICAOFacilityDesignationDesignatorATISCode [212] ICAOFacilityDesignationDesignatorATISCode,

-- [icaofacilitydesignationdesignator] ALTIMETER [altimeter] Urg(N)/Alr(M)/Resp(R)
uM213ICAOFacilityDesignationDesignatorAltimeter [213] ICAOFacilityDesignationDesignatorAltimeter,

-- RUNWAY [runway] VISUAL RANGE [rvr] Urg(N)/Alr(M)/Resp(R)

	uM214RunwayRVR	[214] RunwayRVR,
--	TURN [degrees][direction] uM215DegreesDirection	Urg(N)/Alr(M)/Resp(W/U) [215] DirectionDegrees,
--	REQUEST FLIGHT PLAN uM216NULL	Urg(N)/Alr(M)/Resp(Y) [216] NULL,
--	REPORT ARRIVAL uM217NULL	Urg(N)/Alr(M)/Resp(Y) [217] NULL,
--	REQUEST ALREADY RECEIVED uM218NULL	Urg(L)/Alr(N)/Resp(N) [218] NULL,
--	STOP CLIMB AT [level~~altitude~~] uM219Level~~Altitude~~	Urg(U)/Alr(M)/Resp(W/U) [219] Level~~Altitude~~,
--	STOP DESCENT AT [level~~altitude~~] uM220Level~~Altitude~~	Urg(U)/Alr(M)/Resp(W/U) [220] Level~~Altitude~~,
--	STOPTURN HEADING [degrees] uM221Degrees	Urg(U)/Alr(M)/Resp(W/U) [221] Degrees,
--	NO SPEED RESTRICTION uM222NULL	Urg(L)/Alr(L)/Resp(R) [222] NULL,
--	REDUCE TO MINIMUM APPROACH SPEED uM223NULL	Urg(N)/Alr(M)/Resp(W/U) [223] NULL,
--	NO DELAY EXPECTED uM224NULL	Urg(N)/Alr(L)/Resp(R) [224] NULL,
--	DELAY NOT DETERMINED uM225NULL	Urg(N)/Alr(L)/Resp(R) [225] NULL,
--	EXPECTED APPROACH TIME [time] uM226Time	Urg(N)/Alr(L)/Resp(R) [226] Time,
--	LOGICAL ACKNOWLEDGMENT uM227NULL	Urg(N)/Alr(M)/Resp(N) [227] NULL,
--	REPORT ETA [position] uM228Position	Urg(L)/Alr(L)/Resp(Y) [228] Position,
--	REPORT ALTERNATE AERODROME uM229NULL	Urg(L)/Alr(L)/Resp(Y) [229] NULL,
--	IMMEDIATELY uM230NULL	Urg(D)/Alr(H)/Resp(N) [230] NULL,
--	STATE PREFERRED LEVEL~~ALTITUDE~~ uM231NULL	Urg(L)/Alr(L)/Resp(Y) [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,

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

        ...
        }

-- ------------------------------------------------------------------------
-- Downlink message element
-- ------------------------------------------------------------------------

ATCDownlinkMsgElementId ::= CHOICE
        {
--      WILCO                                       Urg(N)/Alr(M)/Resp(N)
        dM0NULL                                     [0] NULL,

--      UNABLE                                      Urg(N)/Alr(M)/Resp(N)
        dM1NULL                                     [1] NULL,

--      STANDBY                                     Urg(N)/Alr(M)/Resp(N)
        dM2NULL                                     [2] NULL,

--      ROGER                                       Urg(N)/Alr(M)/Resp(N)
        dM3NULL                                     [3] NULL,

--      AFFIRM                                      Urg(N)/Alr(M)/Resp(N)
        dM4NULL                                     [4] NULL,

--      NEGATIVE                                    Urg(N)/Alr(M)/Resp(N)
        dM5NULL                                     [5] NULL,

--      REQUEST [levelaltitude]                     Urg(N)/Alr(L)/Resp(Y)
        dM6LevelAltitude                                [6] LevelAltitude,

--      REQUEST BLOCK [levelaltitude] TO [levelaltitude]    Urg(N)/Alr(L)/Resp(Y)
        dM7LevelAltitudeLevelAltitude                   [7] LevelAltitudeLevelAltitude,

--      REQUEST CRUISE CLIMB TO [levelaltitude]     Urg(N)/Alr(L)/Resp(Y)
        dM8LevelAltitude                                [8] LevelAltitude,

--      REQUEST CLIMB TO [levelaltitude]            Urg(N)/Alr(L)/Resp(Y)
        dM9LevelAltitude                                [9] LevelAltitude,
```

```
--      REQUEST DESCENT TO [levelaltitude]              Urg(N)/Alr(L)/Resp(Y)
        dM10LevelAltitude                                       [10] LevelAltitude,

--      AT [position] REQUEST CLIMB TO [levelaltitude]  Urg(N)/Alr(L)/Resp(Y)
        dM11PositionLevelAltitude                               [11] PositionLevelAltitude,

--      AT [position] REQUEST DESCENT TO [levelaltitude]        Urg(N)/Alr(L)/Resp(Y)
        dM12PositionLevelAltitude                               [12] PositionLevelAltitude,

--      AT [time] REQUEST CLIMB TO [levelaltitude]              Urg(N)/Alr(L)/Resp(Y)
        dM13TimeLevelAltitude                           [13] TimeLevelAltitude,

--      AT [time] REQUEST DESCENT TO [levelaltitude]    Urg(N)/Alr(L)/Resp(Y)
        dM14TimeLevelAltitude                           [14] TimeLevelAltitude,

--      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 [level~~altitude~~] Urg(N)/Alr(L)/Resp(Y)
dM28Level~~Altitude~~ [28] Level~~Altitude~~,

-- CLIMBING TO [level~~altitude~~] Urg(N)/Alr(M)/Resp(N)
dM29Level~~Altitude~~ [29]Level~~Altitude~~,

-- DESCENDING TO [level~~altitude~~] Urg(N)/Alr(M)/Resp(N)
dM30Level~~Altitude~~ [30] Level~~Altitude~~,

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

-- PRESENT LEVEL~~ALTITUDE~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(N)
dM32Level~~Altitude~~ [32] Level~~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 [level~~altitude~~] Urg(N)/Alr(M)/Resp(N)
dM37Level~~Altitude~~ [37] Level~~Altitude~~,

-- ASSIGNED LEVEL~~ALTITUDE~~ [level~~altitude~~] Urg(N)/Alr(M)/Resp(N)
dM38Level~~Altitude~~ [38] Level~~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 [~~beacon~~code]                  Urg(N)/Alr(M)/Resp(N)
    dM47~~Beacon~~Code                          [47] ~~Beacon~~Code,

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

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

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

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

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

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

--  WHEN CAN WE EXPECT CRUISE CLIMB TO [level~~altitude~~]
                                                Urg(L)/Alr(L)/Resp(Y)
    dM54Level~~Altitude~~                       [54] Level~~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 [personsonboard] PERSONS ON BOARD
                                                Urg(U)/Alr(H)/Resp(N)
    dM57RemainingFuelPersonsOnBoard             [57] RemainingFuelPersonsOnBoard,

--  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 [levelaltitude] dM61LevelAltitude	Urg(U)/Alr(H)/Resp(N) [61] LevelAltitude,
--	ERROR [errorInformation] dM62ErrorInformation	Urg(U)/Alr(L)/Resp(N) [62] ErrorInformation,
--	NOT CURRENT DATA AUTHORITY dM63NULL	Urg(L)/Alr(L)/Resp(N) [63] NULL,
--	[icaofacilitydesignationdesignator] dM64ICAOFacilityDesignationDesignator	Urg(L)/Alr(L)/Resp(N) [64] ICAOFacilityDesignationDesignator,
--	DUE TO WEATHER dM65NULL	Urg(L)/Alr(L)/Resp(N) [65] NULL,
--	DUE TO AIRCRAFT PERFORMANCE dM66NULL	Urg(L)/Alr(L)/Resp(N) [66] NULL,
--	[freetext] dM67FreeText	Urg(N)/Alr(L)/Resp(N) [67] FreeText,
--	[freetext] dM68FreeText	Urg(D)/Alr(H)/Resp(Y) [68] FreeText,
--	REQUEST VMC DESCENT dM69NULL	Urg(N)/Alr(L)/Resp(Y) [69] NULL,
--	REQUEST HEADING [degrees] dM70Degrees	Urg(N)/Alr(L)/Resp(Y) [70] Degrees,
--	REQUEST GROUND TRACK [degrees] dM71Degrees	Urg(N)/Alr(L)/Resp(Y) [71] Degrees,
--	REACHING [levelaltitude] dM72LevelAltitude	Urg(N)/Alr(M)/Resp(N) [72] LevelAltitude,
--	[versionnumber] dM73Versionnumber	Urg(L)/Alr(L)/Resp(N) [73] VersionNumber,
--	REQUEST TO MAINTAIN OWN SEPARATION AND VMC dM74NULL	Urg(L)/Alr(L)/Resp(Y) [74] NULL,
--	AT PILOTS DISCRETION dM75NULL	Urg(L)/Alr(L)/Resp(N) [75] NULL,
--	REACHING BLOCK [levelaltitude] TO [levelaltitude] dM76LevelAltitudeLevelAltitude	Urg(N)/Alr(N)/Resp(N) [76] LevelAltitudeLevelAltitude,
--	ASSIGNED BLOCK [levelaltitude] TO [levelaltitude] dM77LevelAltitudeLevelAltitude	Urg(N)/Alr(N)/Resp(N) [77] LevelAltitudeLevelAltitude,
--	AT [time] [distance] [tofrom] [position] dM78TimeDistanceToFromPosition	Urg(N)/Alr(N)/Resp(N) [78] TimeDistanceToFromPosition,

```
--  ATIS [atiscode]                                   Urg(N)/Alr(M)/Resp(N)
    dM79AtisCode                                      [79] ATISCode,

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

--  WE CAN ACCEPT [levelaltitude] AT [time]           Urg(N)/Alr(L)/Resp(N)
    dM81LevelAltitudeTime                             [81] LevelAltitudeTime,

--  WE CANNNOT ACCEPT [levelaltitude]                 Urg(N)/Alr(L)/Resp(N)
    dM82LevelAltitude                                 [82] LevelAltitude,

--  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] [direction] at[time]   Urg(N)/Alr(L)/Resp(N)
    dM85DirectionDistanceOffsetDirectionTime          [85] DistanceOffsetDirectionTime,

--  WE CANNNOT ACCEPT [direction] [distanceOffset] [direction]
                                                      Urg(N)/Alr(L)/Resp(N)
    dM86DirectionDistanceOffsetDirection              [86] DistanceOffsetDirection,

--  WHEN CAN WE EXPECT CLIMB TO [levelaltitude]       Urg(L)/Alr(L)/Resp(Y)
    dM87LevelAltitude                                 [87] LevelAltitude,

--  WHEN CAN WE EXPECT DESCENT TO [levelaltitude]
                                                      Urg(L)/Alr(L)/Resp(Y)
    dM88LevelAltitude                                 [88] LevelAltitude,

--  MONITORING[icaounitname] [frequency]              Urg(L)/Alr(M)/Resp(N)
    dM89IcaoUunitnameFrequency                        [89] ICAOUnitNameFrequency,

--  [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] dM96FreeText	Urg(U)/Alr(L)/Resp(N) [96] FreeText,
--	[freetext] dM97FreeText	Urg(L)/Alr(L)/Resp(N) [97] FreeText,
--	[freetext] dM98FreeText	Urg(N)/Alr(N)/Resp(N) [98] FreeText,
--	CURRENT DATA AUTHORITY dM99NULL	Urg(L)/Alr(L)/Resp(N) [99] NULL,
--	LOGICAL ACKNOWLEDGMENT dM100NULL	Urg(N)/Alr(M)/Resp(N) [100] NULL,
--	REQUEST END OF SERVICE dM101NULL	Urg(L)/Alr(L)/Resp(Y) [101] NULL,
--	LANDING REPORT dM102NULL	Urg(N)/Alr(N)/Resp(N) [102] NULL,
--	CANCELLING IFR dM103NULL	Urg(N)/Alr(N)/Resp(Y) [103] NULL,
--	ETA[position][time] dM104PositionTime	Urg(L)/Alr(L)/Resp(N) [104] PositionTime,
--	ALTERNATE AERODROME[airport] dM105Airport	Urg(L)/Alr(L)/Resp(N) [105] Airport,
--	PREFERRED LEVEL~~ALTITUDE~~[level~~altitude~~] dM106Level~~Altitude~~	Urg(L)/Alr(L)/Resp(N) [106] Level~~Altitude~~,
--	NOT AUTHORIZED NEXT DATA AUTHORITY dM107NULL	Urg(L)/Alr(L)/Resp(N) [107] NULL,
--	DE-ICING COMPLETE dM108NULL	Urg(L)/Alr(L)/Resp(N) [108] NULL,
--	TOP OF DESCENT [time] dM109Time	Urg(L)/Alr(L)/Resp(N), [109] Time,
--	TOP OF DESCENT [position] dM110Position	Urg(L)/Alr(L)/Resp(N), [110] Position,
--	TOP OF DESCENT [time] [position] dM111TimePosition	Urg(L)/Alr(L)/Resp(N), [111] TimePosition,
--	SQUAWKING 7500 dM112NULL	Urg(U)/Alr(H)/Resp(N), [112] NULL,
--	[speedType] [speedType] [speedType] SPEED [speed]	Urg(N)/Alr(L)/Resp(N),

```
        dM113SpeedTtypeSpeedTtypeSpeedTtypeSpeed
                                                    [113] SpeedTtypeSpeedTtypeSpeedTtypeSpeed,
        ...
        }

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

AircraftEquipmentCode ::= SEQUENCE
        {
        approachEquipmentAvailable      [0]     BOOLEAN,
        approachEquipmentStatus         [1]     SEQUENCE SIZE (1..24) 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
        {
        noAltitude              [0]     NULL,
        singleAltitude          [1]     AltitudeType,
        blockAltitude           [2]     SEQUENCE SIZE (2) OF AltitudeType
        }


AltitudeType ::=  CHOICE
        {
        altitudeFeet            [0]     AltitudeFeet,
        altitudeMeters          [1]     AltitudeMeters,
        altitudeFlightLevel     [2]     AltitudeFlightLevel,
```

~~altitudeFlightLevelMetric [3] AltitudeFlightLevelMetric~~
~~}~~

~~**AltitudeAltitude** ::= SEQUENCE SIZE (2) OF Altitude~~

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

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

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

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

~~**AltitudeFeet** ::= INTEGER (-600..70000)~~
~~--unit = Feet, Range (-600..700000), resolution = 10~~

~~**AltitudeMeters** ::= INTEGER (-30..25000)~~
~~--unit = Meter, Range (-30..25000), 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,
        aTWLevelAltitudes               [3]     ATWLevelAltitudeSequence        OPTIONAL
        }

ATWLevelAltitude ::= SEQUENCE
        {
        atw             ATWLevelAltitudeTolerance,
        levelaltitude           LevelAltitude
        }

ATWLevelAltitudeSequence ::= SEQUENCE SIZE (1..2) OF ATWLevelAltitude

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

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

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

BeaconCode ::= SEQUENCE SIZE (4) OF BeaconCodeOctalDigit

BeaconCodeOctalDigit ::= INTEGER (0..7)

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

~~Beacon~~Code ::= SEQUENCE SIZE (4) OF ~~Beacon~~CodeOctalDigit

~~Beacon~~CodeOctalDigit ::= INTEGER (0..7)

~~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 (1..150)
        --unit = Minute, Range (0.1..15.0), resolution = 0.1

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

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

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

DistanceKm ::= INTEGER (0..8000)
--      unit = Kilometer, Range (0..1000), resolution = 0.25

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
        }

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

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

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

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

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

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

ErrorInformation ::=  ENUMERATED
        {
        unrecognizedMsgReferenceNumber          (0),
        endServiceWithPendingMsgs               (1),
        logicalAcknowledgmentNotAccepted        (2),
        moreThanOneNextDataAuthorityElement     (3),
        insufficientResourcesMessageStorageCapacity     (4),
        ...
        }

ICAOFacilityDesignationDesignator ::=  IA5String (SIZE (8))

ICAOFacilityFunction ::= ENUMERATED
        {
        center          (0),
        approach        (1),
        tower           (2),
```

```
        final                   (3),
        groundControl           (4),
        clearanceDelivery       (5),
        departure               (6),
        control                 (7)
        }

~~ICAO~~FacilityDesignation~~Designator~~Altimeter ::= SEQUENCE
        {
        ~~iCAOF~~facilityDesignation~~Designator~~       ~~ICAO~~UnitName,
        altimeter               Altimeter
        }

~~ICAO~~FacilityDesignation~~Designator~~ATISCode ::= SEQUENCE
        {
        ~~iCAOF~~facilityDesignation~~Designator~~       ~~ICAO~~UnitName,
        aTISCode                ATISCode
        }

~~ICAO~~FacilityIdentification ::= CHOICE
        {
        ~~iCAOF~~facilityDesignation~~Designator~~       [0]     ~~ICAO~~FacilityDesignation~~Designator~~,
        ~~iCAOF~~facilityName           [1]     ~~ICAO~~FacilityName
        }

~~ICAO~~FacilityName ::= IA5String (SIZE (3..18))

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,
        frequencyvhfchannel     [4]     Frequencyvhfchannel
        }

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..17000)
-- unit = Megahertz, Range (117.00000..138.00000), resolution = 0.0250.00833

Frequencyvhfchannel ::= INTEGER (23600..27600)
-- Range (118.000..138.000), resolution = 0.005

FurtherInstructions ::= SEQUENCE
{

beaconCcode	[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,
aTWlevelaltitude	[2]	ATWLevelAltitude	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,	
levelaltitude	[1]	LevelAltitude,	
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)
        -- unit = Nautical Mile, Range (0..50), resolution = 1

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

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

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

Level~~Altitude~~ ::= CHOICE
{
noLevel~~Altitude~~ [0] NULL,
singleLevel~~Altitude~~ [1] Level~~Altitude~~Type,
blockLevel~~Altitude~~ [2] SEQUENCE SIZE (2) OF Level~~Altitude~~Type
}

Level~~Altitude~~Type ::= CHOICE
{
level~~altitude~~Feet [0] Level~~Altitude~~Feet,
level~~altitude~~Meters [1] Level~~Altitude~~Meters,
level~~altitude~~FlightLevel [2] Level~~Altitude~~FlightLevel,
level~~altitude~~FlightLevelMetric [3] Level~~Altitude~~FlightLevelMetric
}

Level~~Altitude~~Level~~Altitude~~ ::= SEQUENCE SIZE (2) OF Level~~Altitude~~

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

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

Level~~Altitude~~Position ::= SEQUENCE
{
level~~altitude~~ Level~~Altitude~~,
position Position
}

Level~~Altitude~~ProcedureName ::= SEQUENCE
{
level~~altitude~~ Level~~Altitude~~,
procedureName ProcedureName
}

Level~~Altitude~~Feet ::= INTEGER (-600..70000)
--unit = Feet, Range (-600..700000), resolution = 10

Level~~Altitude~~Meters ::= INTEGER (-30..25000)
--unit = Meter, Range (-30..25000), resolution = 1

Level~~Altitude~~Speed ::= SEQUENCE
{

```
        levelaltitude                   LevelAltitude,
        speed               Speed
        }

LevelAltitudeSpeedSpeed ::= SEQUENCE
        {
        levelaltitude                   LevelAltitude,
        speeds              SpeedSpeed
        }

LevelAltitudeTime ::= SEQUENCE
        {
        levelaltitude                   LevelAltitude,
        time                Time
        }

LevelsOfFlight ::= CHOICE
        {
        levelaltitude                   [0]     LevelAltitude,
        procedureName       [1]     ProcedureName,
        levelaltitudeProcedureName      [2]     LevelAltitudeProcedureName
        }

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

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

PointLevel~~Altitude~~ ::= SEQUENCE
{
level~~altitude~~FlightLevel [0] Level~~Altitude~~FlightLevel,
aTWLevel~~Altitude~~Tolerance [1] ATWLevel~~Altitude~~Tolerance OPTIONAL
}

PointLevel~~Altitude~~Block ::= SEQUENCE SIZE (2) OF Level~~Altitude~~FlightLevel

PointDetail ::= SEQUENCE
{
latitudeLongitude [0] LatitudeLongitude,
fixName [1] FixName OPTIONAL,
truetrackangle [2] DegreesTrue OPTIONAL,
distancetonextpoint [3] DistanceToNextPoint OPTIONAL,
pointLevel~~Altitude~~ [4] PointLevel~~Altitude~~ OPTIONAL,
pointLevel~~Altitude~~Block [5] PointLevel~~Altitude~~Block OPTIONAL
}

Position ::= CHOICE
{
fixName [0] FixName,
navaid [1] Navaid,
airport [2] Airport,
latitudeLongitude [3] LatitudeLongitude,
placeBearingDistance [4] PlaceBearingDistance
}

PositionLevel~~Altitude~~ ::= SEQUENCE
{
position Position,
level~~altitude~~ Level~~Altitude~~
}

PositionLevelAltitudeLevelAltitude ::= SEQUENCE
{
position Position,
levelaltitudes LevelAltitudeLevelAltitude
}

PositionLevelAltitudeSpeed ::= SEQUENCE
{
positionlevelaltitude PositionLevelAltitude,
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,	
levelaltitude	[2]	LevelAltitude,	
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,
        reportedWaypointLevelAltitude           [21]    LevelAltitude       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
        }

PositionTimeLevelAltitude ::= SEQUENCE
        {
        positionTime            PositionTime,
        levelaltitude                   LevelAltitude
        }

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

Procedure ::= IA5String (SIZE (1..20))

ProcedureName ::= SEQUENCE
        {
        type                    [0]     ProcedureType,
```

```
        procedure               [1]     Procedure,
        transition              [2]     ProcedureTransition     OPTIONAL
        }

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

ProcedureType ::= ENUMERATED
        {
        arrival                 (0),
        approach                (1),
        departure               (2)
        }

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

RemainingFuel ::= Time

RemainingFuelPersonsOnBoard ::=  SEQUENCE
        {
        remainingFuel    RemainingFuel,
        personsOnBoard   PersonsOnBoard
        }

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

RevisionNumber ::= INTEGER (1..16)

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

RouteClearance ::= SEQUENCE
        {
        airportDeparture        [0]     Airport                 OPTIONAL,
        airportDestination      [1]     Airport                 OPTIONAL,
        runwayDeparture         [2]     Runway                  OPTIONAL,
        procedureDeparture      [3]     ProcedureName           OPTIONAL,
        runwayArrival           [4]     Runway                  OPTIONAL,
        procedureApproach       [5]     ProcedureName           OPTIONAL,
        procedureArrival        [6]     ProcedureName           OPTIONAL,
        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,
        waypointSpeedLevelAltitudes             [4]     SEQUENCE SIZE (1..32) OF
WaypointSpeedLevelAltitude      OPTIONAL,
        rTARequiredTimeArrivals         [5]     SEQUENCE SIZE (1..32) OF RTARequiredTimeArrival

OPTIONAL
        }

RTARequiredTimeArrival ::= SEQUENCE
        {
        position                [0]     Position,
        rTATime                 [1]     RTATime,
        rTATolerance            [2]     RTATolerance            OPTIONAL
        }

RTATime ::= SEQUENCE
        {
        time                    Time,
        timeTolerance           TimeTolerance
        }

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

Runway ::= SEQUENCE
        {
        direction               RunwayDirection,
        configuration           RunwayConfiguration
        }

RunwayDirection ::= INTEGER (1..36)
```

```
RunwayConfiguration ::= ENUMERATED
        {
        left            (0),
        right           (1),
        center          (2),
        none            (3)
        }

RunwayRVR ::= SEQUENCE
        {
        runway                  Runway,
        rVR                     RVR
        }

RVR ::= CHOICE
        {
        rVRFeet         [0]     RVRFeet,
        rVRMeters       [1]     RVRMeters
        }

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

RVRMeters ::= INTEGER (0..1500)
        -- unit = Meters (0..1500), resolution = 1

SecondsLatLon ::= INTEGER (0..59)
        --unit = Second, Range (0.. 59), resolution = 1

Speed ::=  CHOICE
        {
        speedIndicated          [0]     SpeedIndicated,
        speedIndicatedMetric    [1]     SpeedIndicatedMetric,
        speedTrue               [2]     SpeedTrue,
        speedTrueMetric         [3]     SpeedTrueMetric,
        speedGround             [4]     SpeedGround,
        speedGroundMetric       [5]     SpeedGroundMetric,
        speedMach               [6]     SpeedMach

        }

SpeedIndicated ::= INTEGER (0..400)
        -- unit = Knots, Range (0..400), resolution = 1

SpeedIndicatedMetric ::= INTEGER (0..800)
        -- unit = Kilometers/Hour, Range (0..800), resolution = 1

SpeedGround ::= INTEGER (-50..2000)
        -- unit = Knots, Range (-50..2000), resolution = 1

SpeedGroundMetric ::= INTEGER (-100..4000)
        -- unit = Kilometers/Hour, Range (-100..4000), resolution = 1
```

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

SpeedSpeed ::= SEQUENCE SIZE (2) OF Speed

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

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

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

SpeedType ::= ENUMERATED
{
noneSpecified (0),
indicated (1),
true (2),
ground (3),
mach (4),
approach (5),
cruise (6),
minimum (7),
maximum (8),
...
}

SpeedTypeSpeedTypeSpeedType ::= SEQUENCE SIZE (3) OF SpeedType

SpeedTypeSpeedTypeSpeedTypeSpeed ::= SEQUENCE
{
speedTypes SpeedTypeSpeedTypeSpeedType,
speed Speed
}

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

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

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

TimeLevel~~Altitude~~ ::= SEQUENCE
{
time Time,
level~~altitude~~ Level~~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

Time~~ICAO~~UnitNameFrequency ::= SEQUENCE
{
time Time,
~~iCAO~~u~~U~~nitName ~~ICAO~~UnitName,
frequency Frequency
}

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

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

TimePositionLevel~~Altitude~~ ::= SEQUENCE
{
timeposition TimePosition,
level~~altitude~~ Level~~Altitude~~
}

TimePositionLevel~~Altitude~~Speed ::= SEQUENCE
{
timeposition TimePosition,
level~~altitude~~speed Level~~Altitude~~Speed
}

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 ::= ENUMERATED
        {
        noneSpecified           (0),
        oppositeDirection       (1),
        sameDirection           (2),
        converging              (3),
        crossing                (4)
        }

Turbulence ::= ENUMERATED
        {
        light           (0),
        moderate        (1),
        severe          (2)
```

```
        }

~~ICAO~~UnitName ::=   SEQUENCE
        {
        ~~iCAOF~~facilityId             ~~ICAO~~FacilityIdentification,
        ~~iCAOF~~facilityFunction       ~~ICAO~~FacilityFunction
        }

~~ICAO~~UnitNameFrequency ::=  SEQUENCE
        {
        ~~iCAOU~~unitName               ~~ICAO~~UnitName,
        frequency                       Frequency
        }


VersionNumber ::=  INTEGER (0..15)

VerticalChange ::= SEQUENCE
        {
        direction                       _____VerticalDirection,
        rate                            VerticalRate
        }

VerticalDirection ::= ENUMERATED
        {
        up      (0),
        down    (1)
        }

VerticalRate ::=  CHOICE
        {
        verticalRateEnglish             [0]     VerticalRateEnglish,
        verticalRateMetric              [1]     VerticalRateMetric
        }

VerticalRateEnglish ::= INTEGER (0..3000)
        -- unit = Feet/Minute, Range (0..30000), resolution = 10

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

WaypointSpeed~~Level~~~~Altitude~~ ::= SEQUENCE
        {
        position                        [0]     Position,
        speed                           [1]     Speed                   OPTIONAL,
        aTWLevel~~Altitude~~s           [2]     ATWLevel~~Altitude~~Sequence    OPTIONAL
        }

WindDirection ::= INTEGER (1..360)
        -- unit = Degree, Range (1..360), resolution = 1

Winds ::= SEQUENCE
        {
```

```
        direction               WindDirection,
        speed                   WindSpeed
        }

WindSpeed ::= CHOICE
        {
        windSpeedEnglish    [0]     WindSpeedEnglish,
        windSpeedMetric     [1]     WindSpeedMetric
        }

WindSpeedEnglish ::= INTEGER (0..255)
        -- unit = Knot, Range (0..255), resolution = 1

WindSpeedMetric ::= INTEGER (0..511)
        -- unit = Kilometer/Hour, Range (0..511), resolution = 1

Year ::= INTEGER (1996..2095)0..99)
        -- unit = Year, Range (1996..2095), resolution = 1


END
```

2.3.5. PROTOCOL DEFINITION

2.3.5.1 Sequence Rules

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

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

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

Note 3.— Primitives are processed in the order received. See 4.4.3.

~~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.*~~

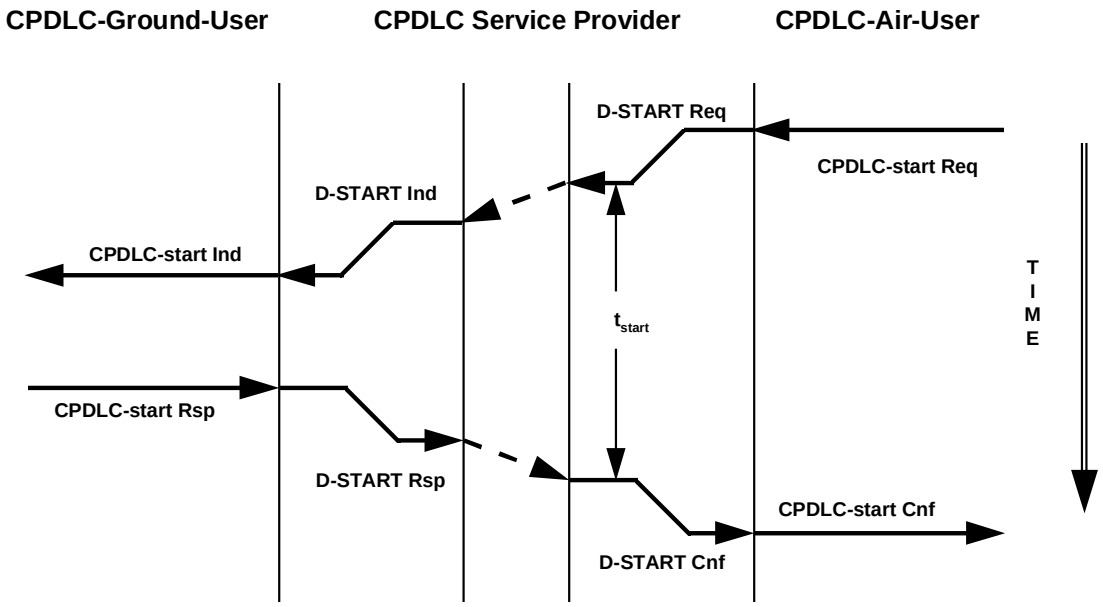

Figure 2.3.5-1. Sequence Diagram for CPDLC-start Service Air Initiated

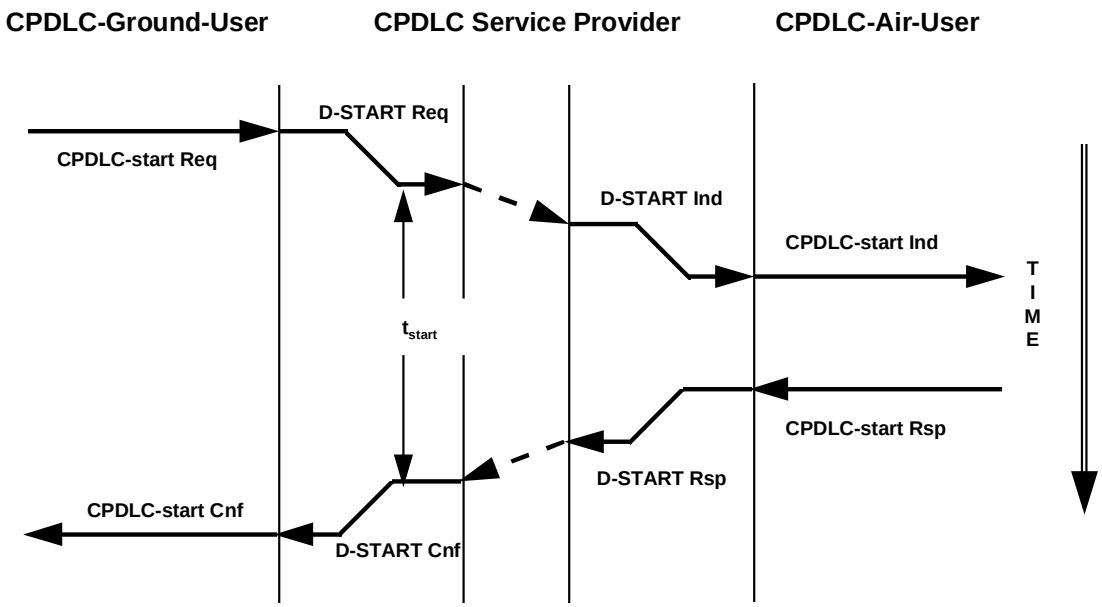

Figure 2.3.5-2. Sequence Diagram for CPDLC-start Service Ground Initiated

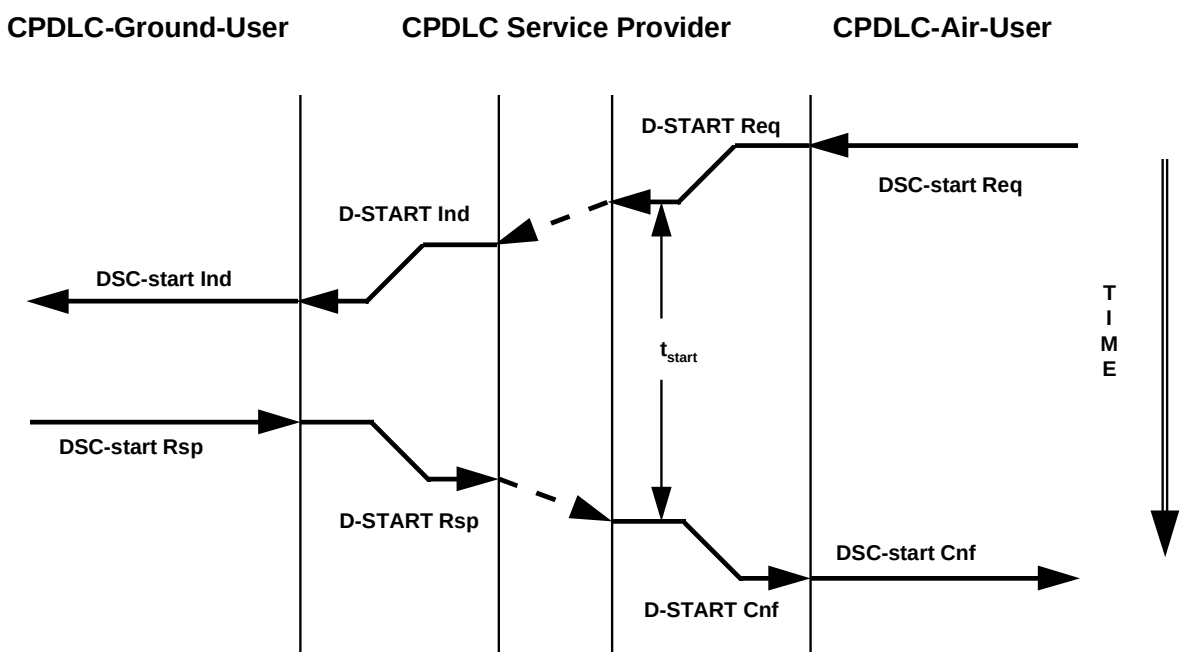

Figure 2.3.5-3. Sequence Diagram for DSC-start Service

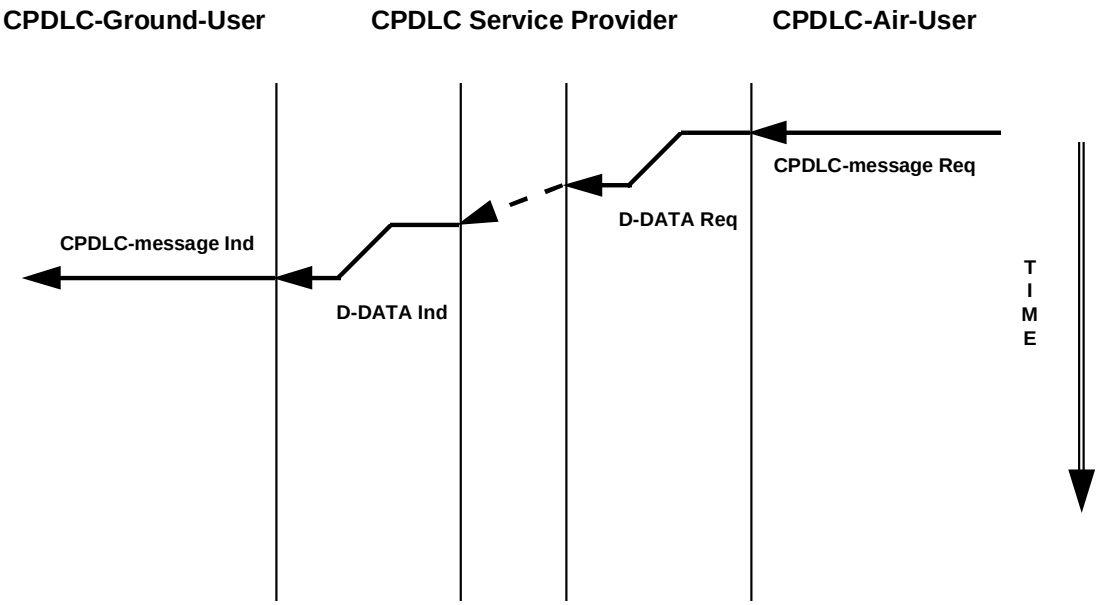

Figure 2.3.5-4. Sequence Diagram for CPDLC-message Service Air Initiated

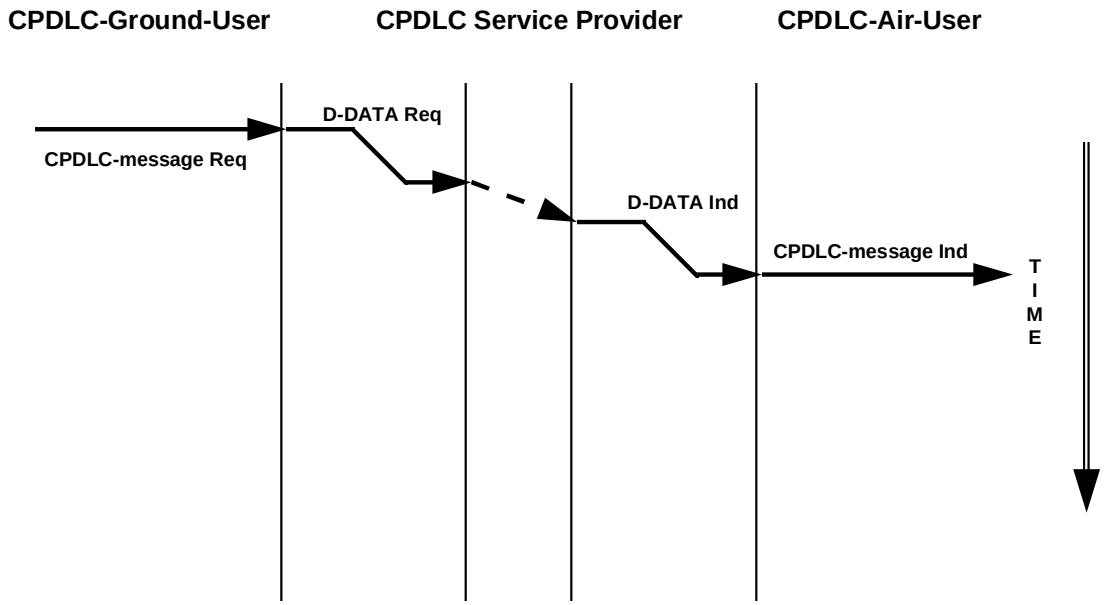

Figure 2.3.5-5. Sequence Diagram for CPDLC-message Service Ground Initiated

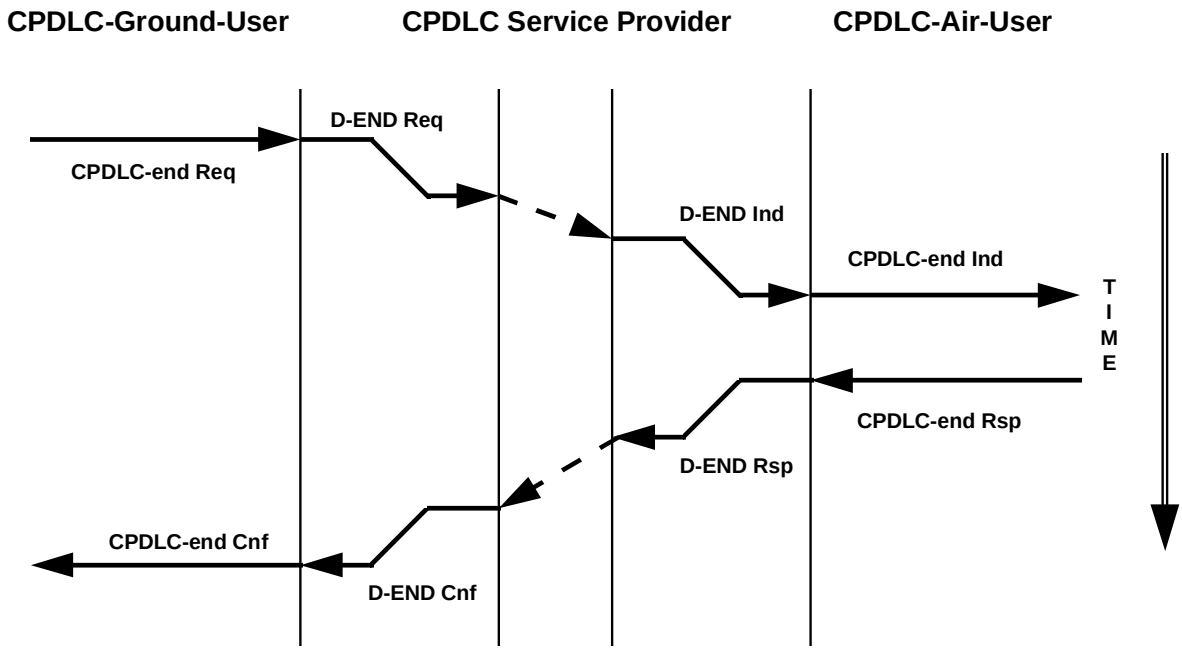

Figure 2.3.5-6. Sequence Diagram for CPDLC-end Service

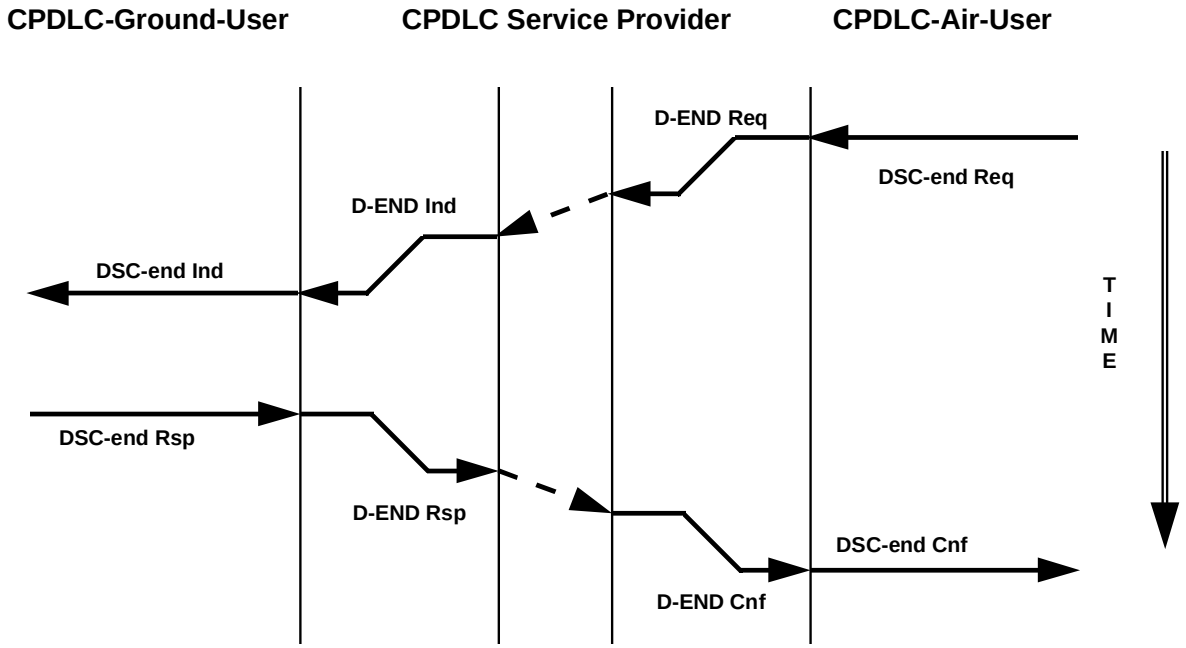

Figure 2.3.5-7. Sequence Diagram for DSC-end Service

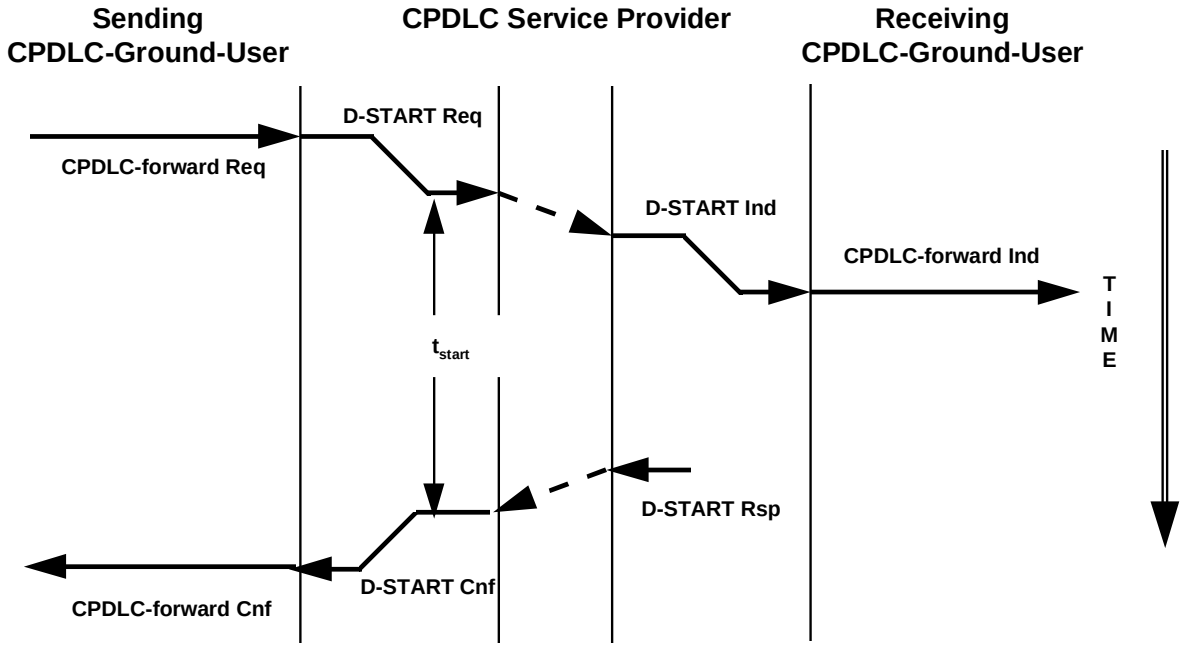

Figure 2.3.5-8. Sequence Diagram for CPDLC-forward Service Ground Forwarding Supported, ASE Version Numbers the Same

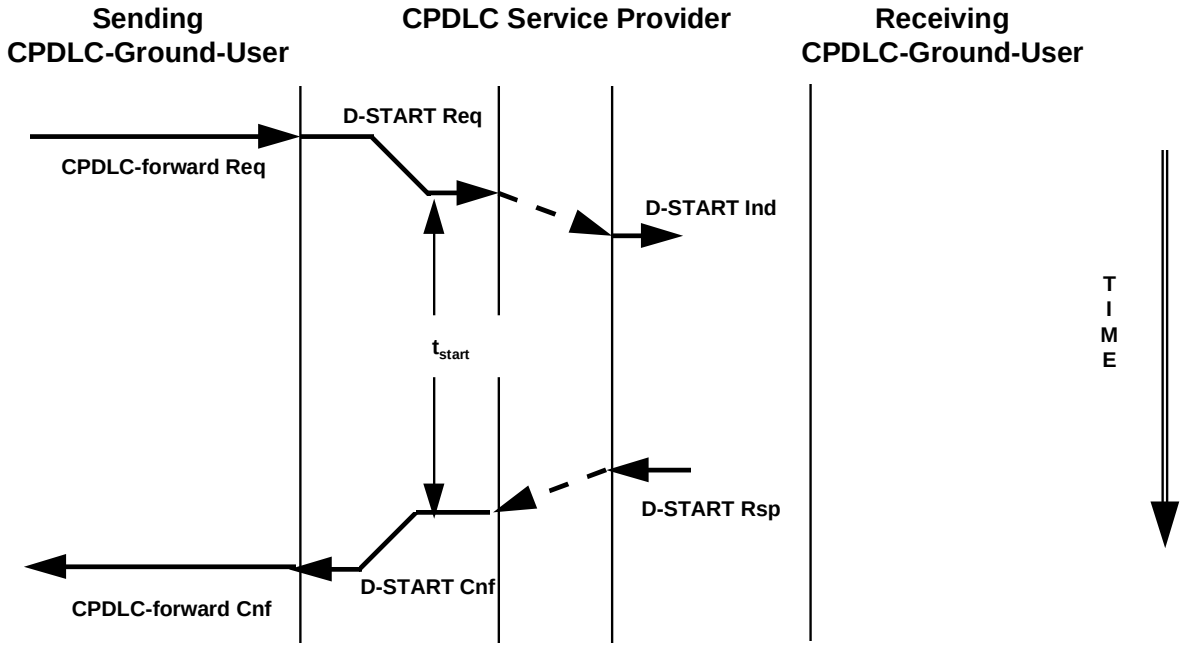

Figure 2.3.5-9. Sequence Diagram for CPDLC-forward Service
Ground Forwarding Not Supported, or
Ground Forwarding Supported and ASE Version Numbers Not the Same

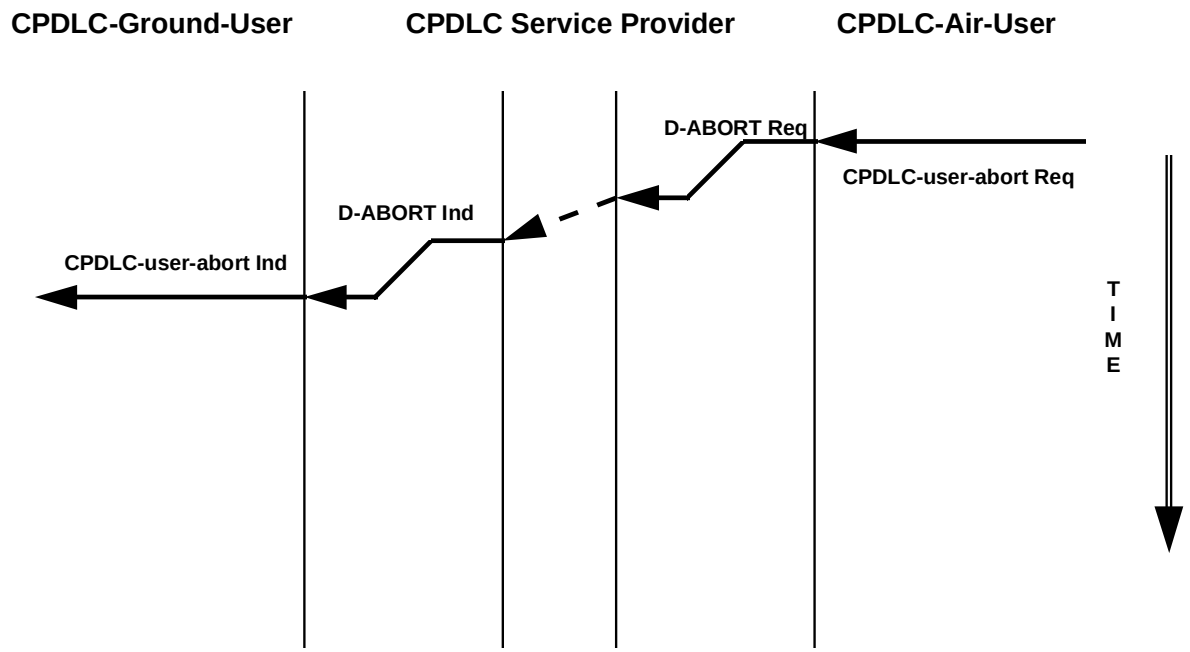

Figure 2.3.5-10. Sequence Diagram for CPDLC-user-abort Service
CPDLC-Air-User Initiated

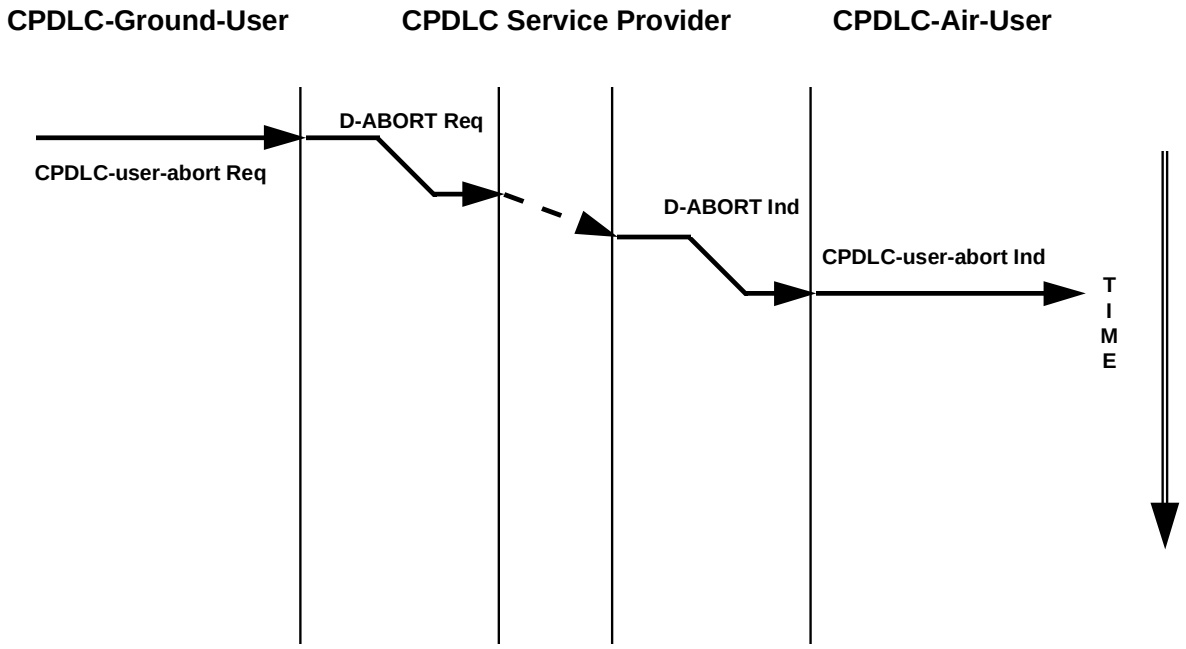

Figure 2.3.5-11. Sequence Diagram for CPDLC-user-abort Service
CPDLC-Ground-User Initiated

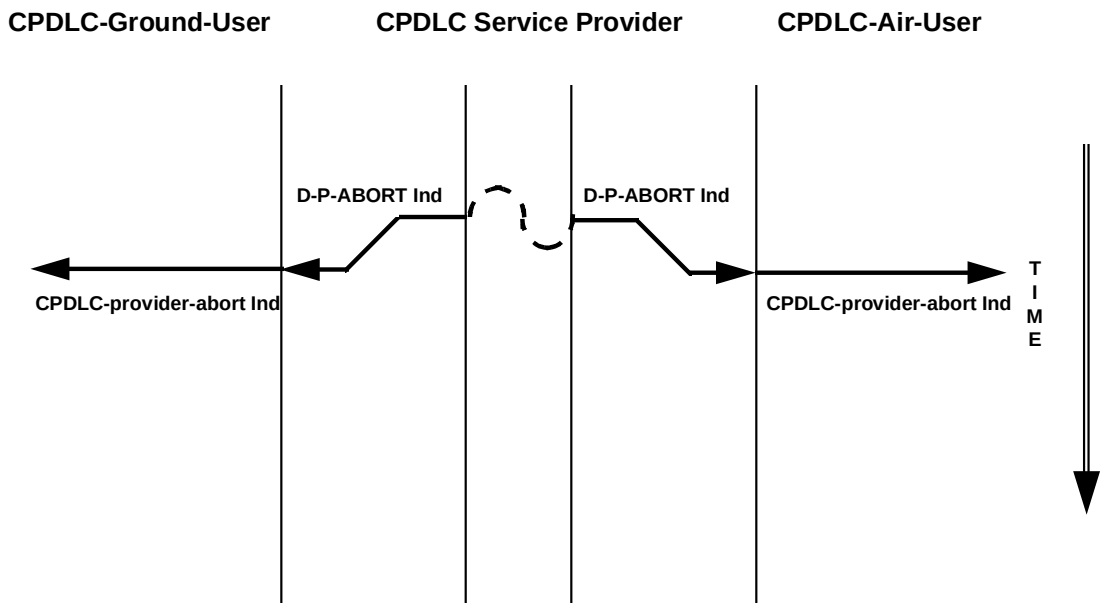

Figure 2.3.5-12. Sequence Diagram for CPDLC-provider-abort Service Dialogue Service Abort

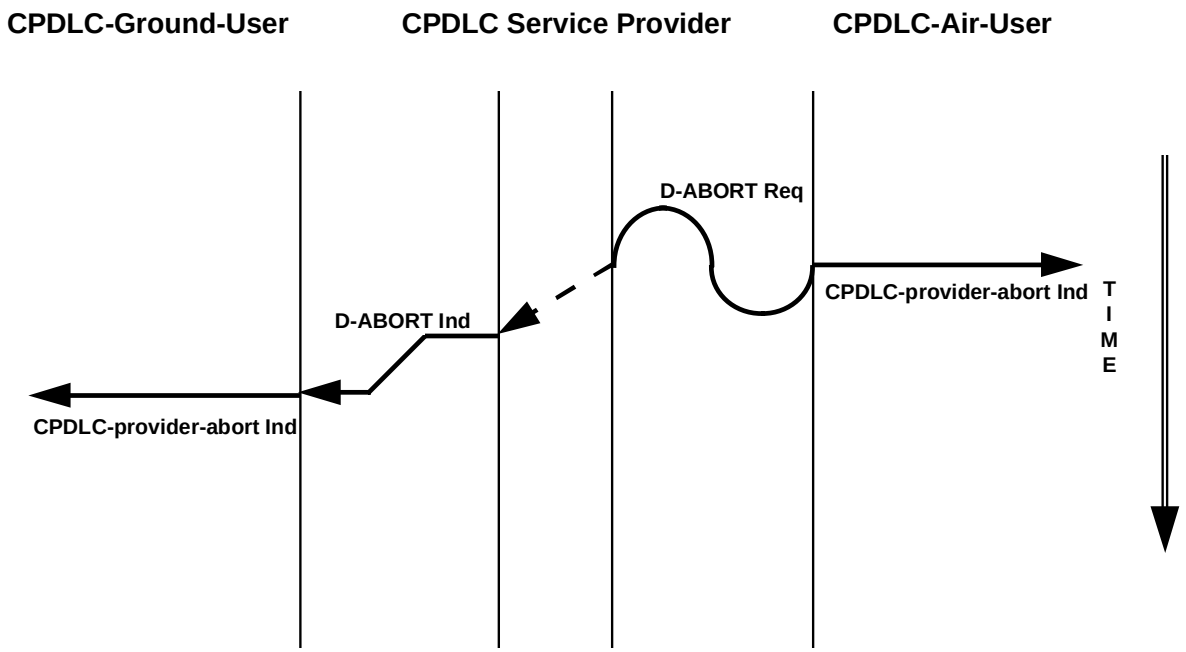

Figure 2.3.5-13. Sequence Diagram for CPDLC-provider-abort Service CPDLC-Air-ASE Abort

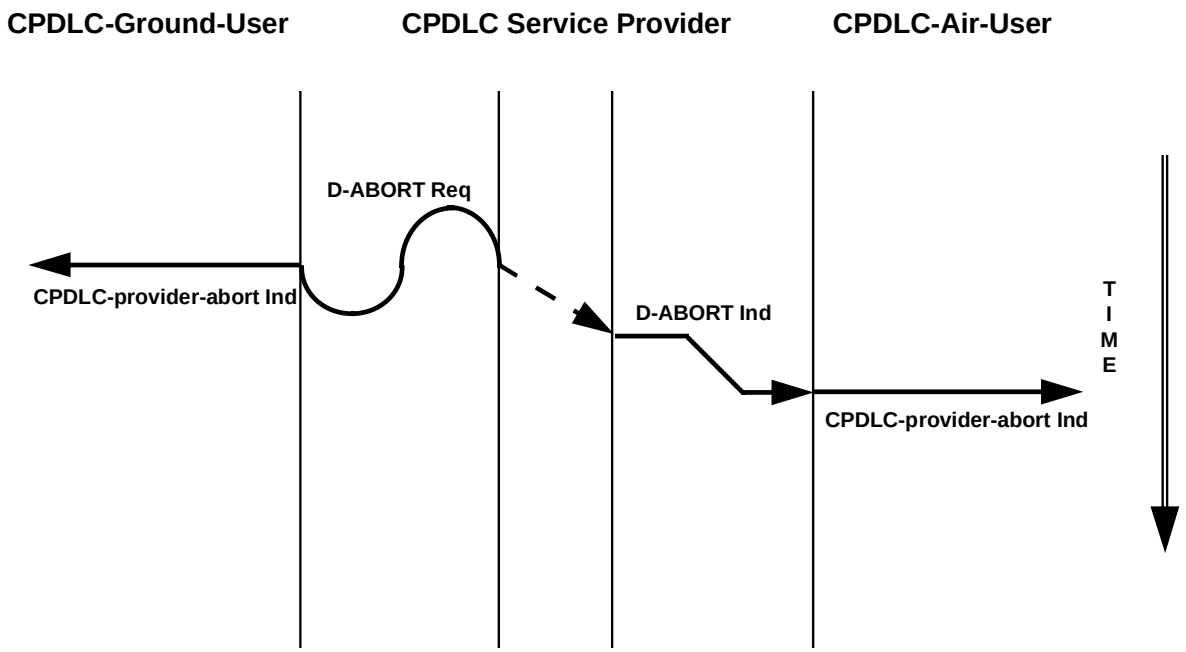

Figure 2.3.5-14. Sequence Diagram for CPDLC-provider-abort Service CPDLC-Ground-ASE Abort

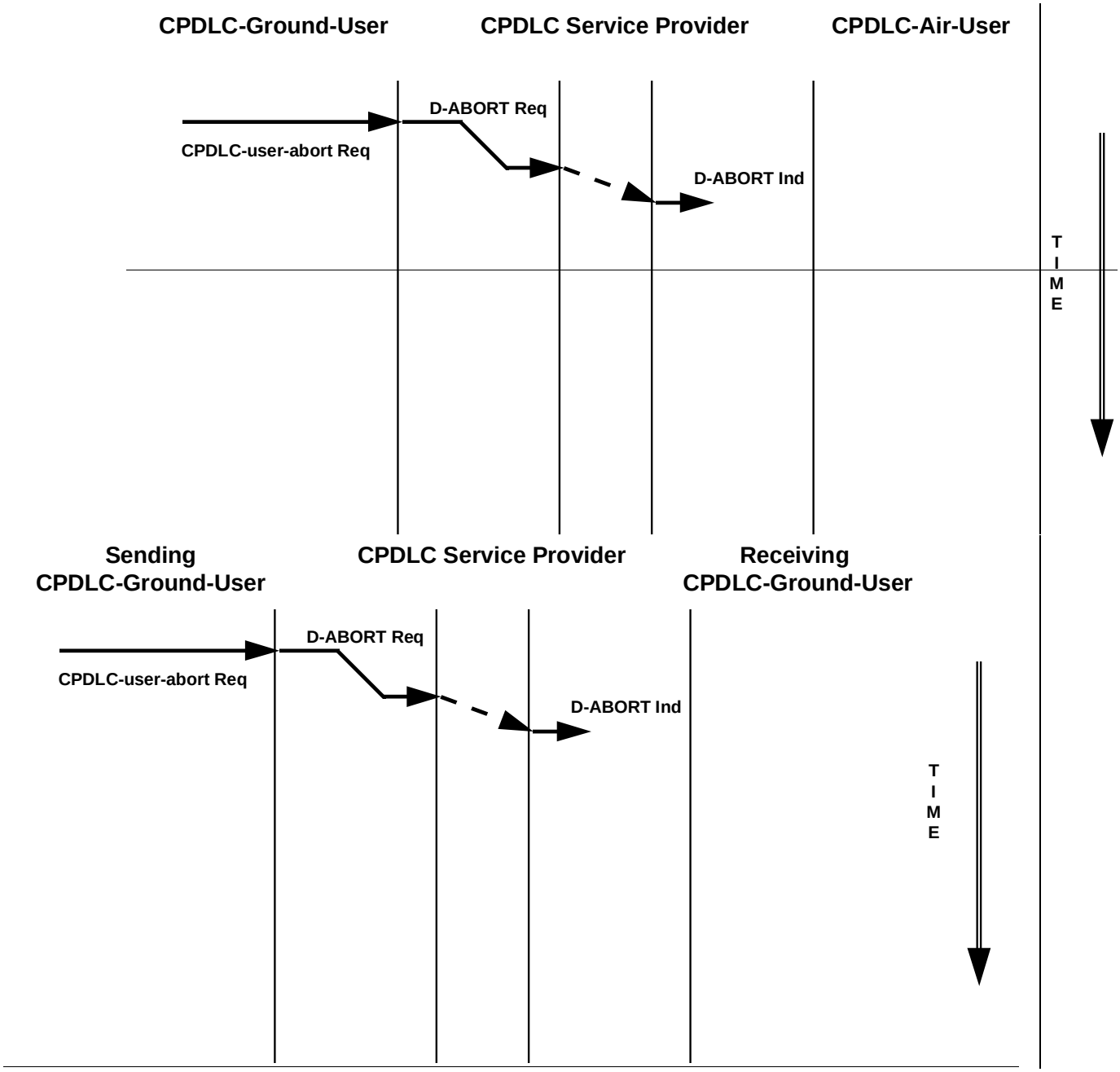

Figure 2.3.5-15. Sequence Diagram for CPDLC-user-abort Service
Sending CPDLC-Ground-User Initiated

.

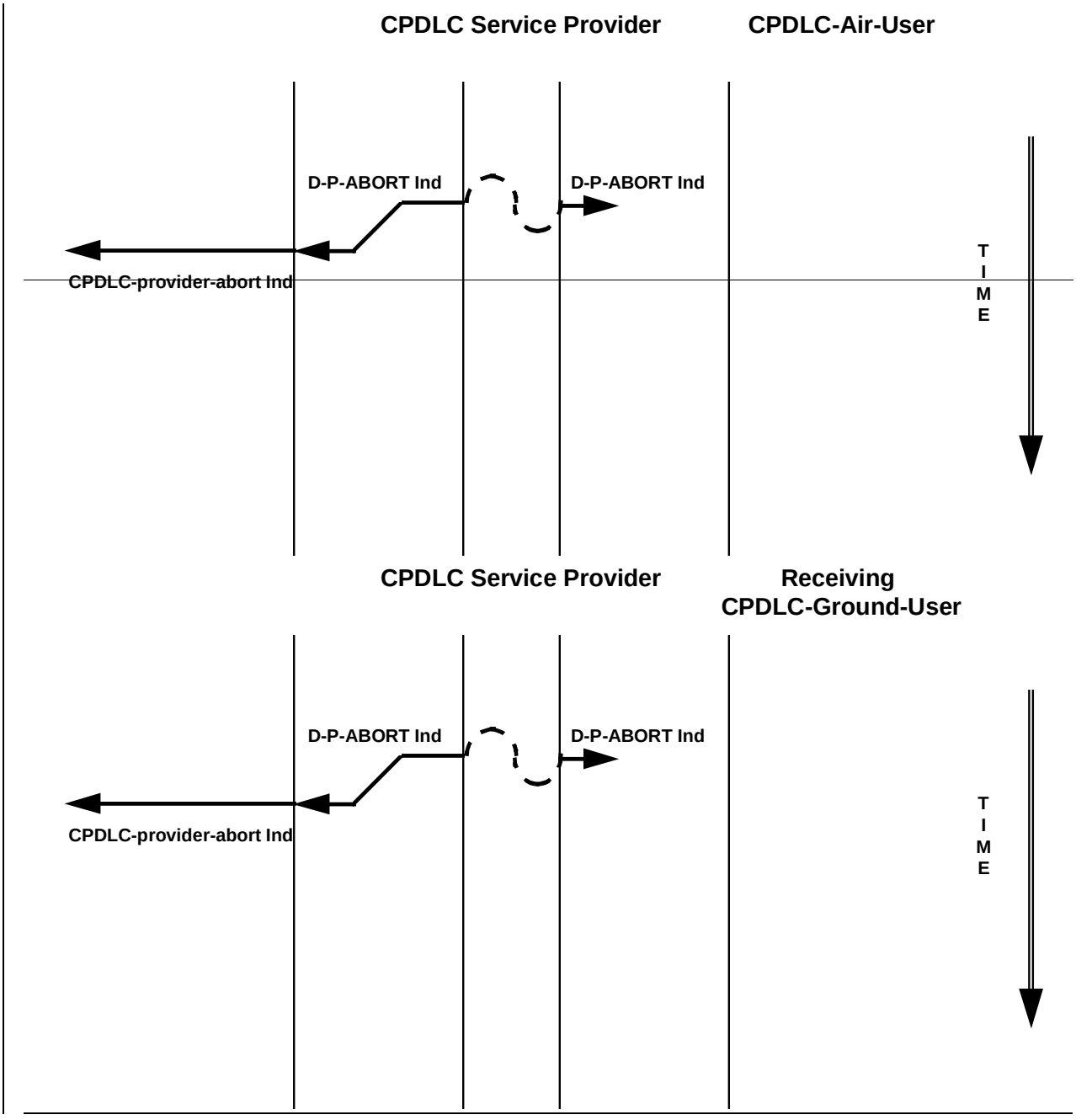

Figure 2.3.5-16. Sequence Diagram for CPDLC-provider-abort Service
Dialogue Service Abort

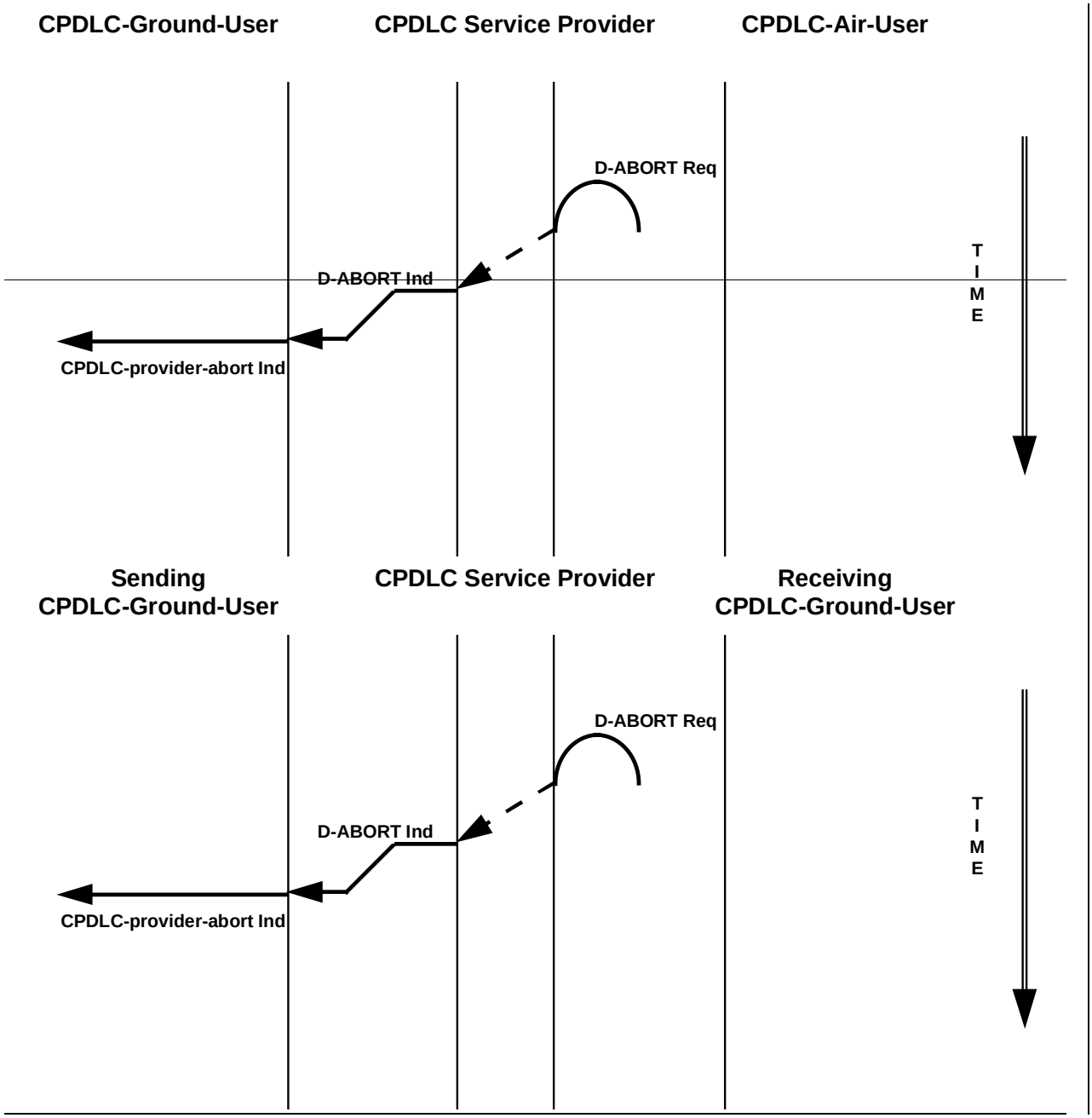

Figure 2.3.5-17. Sequence Diagram for CPDLC-provider-abort Service Receiving CPDLC-Ground-ASE Abort

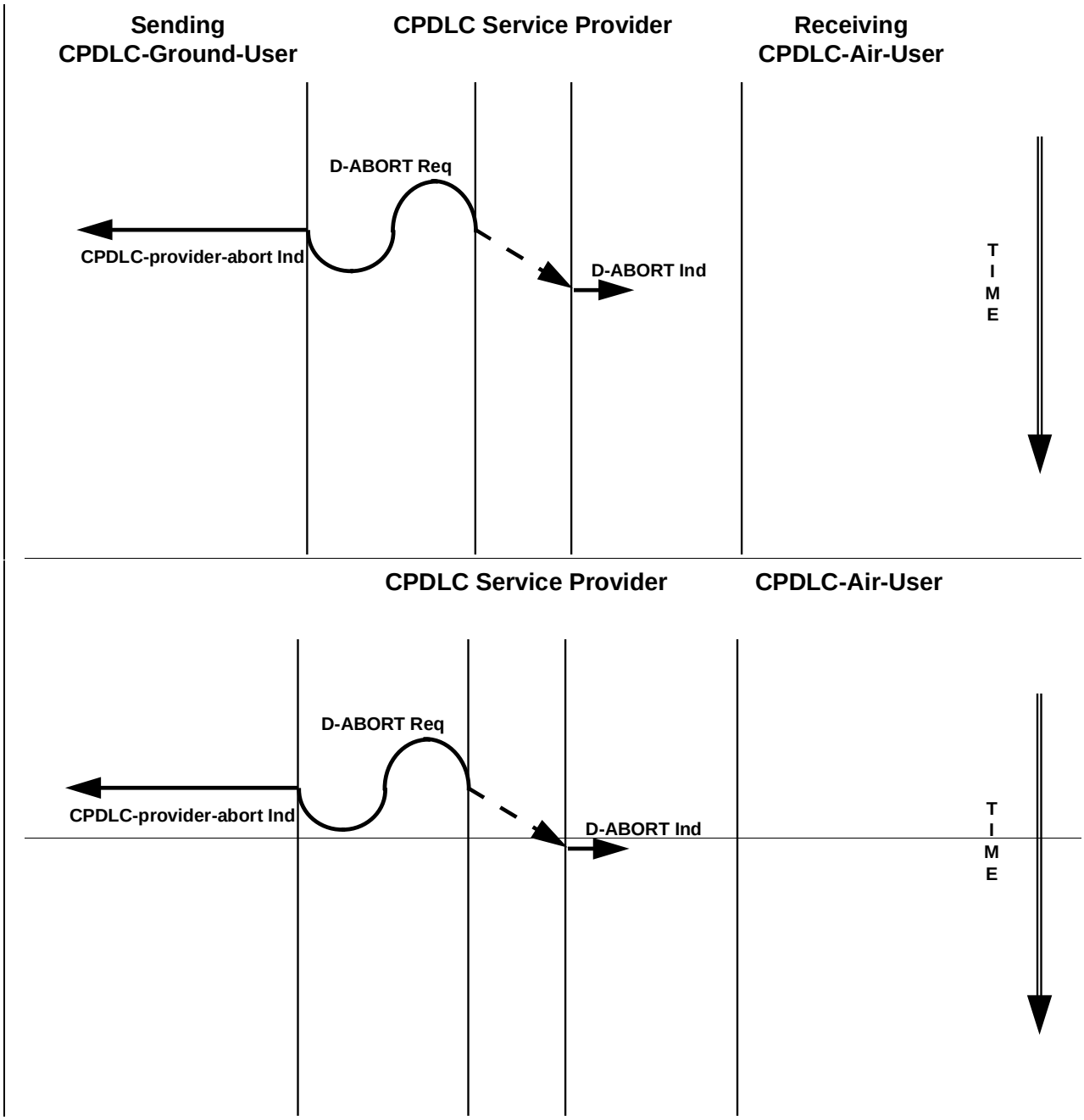

Figure 2.3.5-18. Sequence Diagram for CPDLC-provider-abort Service Sending CPDLC-Ground-ASE Abort

2.3.5.2 CPDLC Service Provider Timers

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

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

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

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

Table 2.3.5-1. CPDLC Service Provider Timers

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

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

2.3.5.3 CPDLC-Air-ASE Protocol Description

2.3.5.3.1 Introduction

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

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

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

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

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

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

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

I. the time it has invoked the CPDLC-start service request until:
A. receipt of a CPDLC-start service confirmation with Result parameter equal to the abstract value "rejected", or
B. invocation of a CPDLC-end service response with the Result parameter set to the abstract value "accepted", or

C. invocation of a CPDLC-user-abort service request, or
D. receipt of CPDLC-user-abort service indication, or
E. receipt of a CPDLC-provider-abort service indication; or

II. the time it has received the CPDLC-start service indication until:
A. invocation of a CPDLC-start service response with Result parameter equal to the abstract value "rejected", or
B. invocation of a CPDLC~~CPDC~~-end service response with the Result parameter set to the abstract value "accepted", or
C. invocation of a CPDLC-user-abort service request, or
D. receipt of CPDLC-user-abort service indication, or
E. receipt of a CPDLC-provider-abort service indication; or

III. the time it has invoked the DSC-start service request until:
A. receipt of a DSC-start service confirmation with Result parameter equal to the abstract value "rejected", or
B. receipt of a DSC-end service confirmation with Result parameter equal to the abstract value "accepted", or
C. invocation of a CPDLC-user-abort service request, or
D. receipt of CPDLC-user-abort service indication, or
E. receipt of a CPDLC-provider-abort service indication.

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

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

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

2.3.5.3.2 **D-START Indication**

2.3.5.3.2.1 Upon receipt of a D-START indication, if the CPDLC-air-ASE is in the *IDLE* state and the D-START *User Data* parameter contains a GroundPDUs [UplinkMessage] APDU, the CPDLC-air-ASE shall:

I. Invoke CPDLC-start service indication containing the following:
A. the D-START *Calling Peer ID* parameter value as the CPDLC-start service *Calling Peer Identifier* parameter value,
B. the D-START *QOS Routing Class* parameter value as the CPDLC-start service *Class of Communication* parameter value,
C. if the GroundPDUs [UplinkMessage] APDU contained in the D-START *User Data* parameter is an ATCUplinkMessage, set the GroundPDUs APDU-element as the CPDLC-start service *CPDLC Message* parameter value, and
II. Enter the *START-IND* state.

2.3.5.3.3 **D-START Confirmation**

2.3.5.3.3.1 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "accepted" and DSC has the abstract value "false" and D-START *User Data* parameter is not provided, the CPDLC-air-ASE shall:

I. Stop timer t_{start},
II. Invoke CPDLC-start service confirmation with the abstract value "accepted" as the CPDLC-start service *Result* parameter value,
III. Enter the *DIALOGUE* state.

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

parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

I. Stop timer t_{start},
II. Invoke CPDLC-start service confirmation containing the following:
    A. if the D-START *User Data* parameter is provided, the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, and
    B. the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and
III. Enter the *IDLE* state.

2.3.5.3.3.3 Upon receipt of a D-START confirmation, if the CPDLC-air-ASE is in the *START-REQ* state and the D-START *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true" and D-START *User Data* parameter is not provided, the CPDLC-air-ASE shall:

I. Stop timer t_{start},
II. Invoke DSC-start service confirmation with the abstract value "accepted" as the DSC-start service *Result* parameter value,
III. Enter the *DIALOGUE* state.

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

I. Stop timer t_{start},
II. Invoke DSC-start service confirmation containing the following:
    A. if the D-START *User Data* parameter is provided, the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, and
    B. the abstract value "rejected" as the DSC-start service *Result* parameter value,
III. Set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

2.3.5.3.4 **D-DATA Indication**

2.3.5.3.4.1 Upon receipt of a D-DATA indication, if the CPDLC-air-ASE is in the *DIALOGUE* state and the APDU contained in the D-DATA *User Data* parameter is a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

I. 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
II. Remain in the *DIALOGUE* state.

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

I. 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
II. Remain in the *END* state.

2.3.5.3.5 **D-END Indication**

2.3.5.3.5.1 Upon receipt of a D-END indication, if the CPDLC-air-ASE is in the *DIALOGUE* state, and DSC has the abstract value "false", and if the D-END *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

I. Invoke CPDLC-end service indication with the APDU contained in the D-END *User Data* parameter as the CPDLC-message service *CPDLC Message* parameter value, if provided, as the CPDLC-end service *CPDLC Message* parameter value, and

II. Enter the *END* state

2.3.5.3.6 **D-END Confirmation**

2.3.5.3.6.1 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the abstract the D-END *Result* parameter has the abstract value "accepted" and DSC has the abstract value "true" and if the D-END *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

I. Invoke DSC-end service confirmation with:
   A. if the D-END *User Data* parameter is provided, the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, and
   B. the abstract value "accepted" as the CPDLC-end service *Result* parameter value,
II. Set DSC to the abstract value "false", and
III. Enter the *IDLE* state.

2.3.5.3.6.2 Upon receipt of a D-END confirmation, if the CPDLC-air-ASE is in the *END* state and the D-END *Result* parameter has the abstract value "rejected", and DSC has the abstract value "true", and if the D-END *User Data* parameter is provided, the *User Data* parameter contains a GroundPDUs [ATCUplinkMessage] APDU, the CPDLC-air-ASE shall:

I. Invoke DSC-end service confirmation with:
   A. if the D-END *User Data* parameter is provided, the APDU contained in the D-END *User Data* parameter as the DSC-end service *CPDLC Message* parameter value, and
   B. the abstract value "rejected" as the CPDLC-end service *Result* parameter value:
II. Enter the *DIALOGUE* state.

2.3.5.3.7 **CPDLC-start Service Request**

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

I. Create an AircraftPDUs APDU with a StartDownMessage APDU element containing:
   A. the abstract value "cpdlc" as the mode,
   B. if provided, the *CPDLC Message* parameter as the DownlinkMessage, or
   C. else, NULL as the DownlinkMessage,
II. Invoke D-START request with the following:
   A. the CPDLC-start service *Called Peer Identifier* parameter value as the D-START *Called Peer ID* parameter value,
   B. the CPDLC-start service *Calling Peer Identifier* parameter value as the D-START *Calling Peer ID* parameter value,
   C. the D-START *Quality of Service* parameters set as follows:
      1. if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START *QOS Routing Class* parameter value, or
      2. The abstract value of "high~~normal~~ priority flight safety messages", as the D-START *QOS Priority* parameter value, and
      3. The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and
   D. the APDU as the D-START *User Data* parameter value;
III. Start timer t_{start}, and
IV. Enter the *START-REQ* state.

2.3.5.3.8 **CPDLC-start Service Response**

2.3.5.3.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-air-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "accepted" and the CPDLC-start service *Reject Reason* parameter is not provided, and DSC has the abstract value "false", the CPDLC-air-ASE shall:

I. Invoke D-START response with the abstract value “accepted” as the D-START *Result* parameter value, and
II. Enter the *DIALOGUE* state.

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

I. If the CPDLC-start service *Reject Reason* parameter is provided, create an AircraftPDUs APDU with an ATCDownlinkMessage APDU element based on the *Reject Reason* parameter,
II. Invoke D-START response with the following:
    A. if created, the APDU as the D-START *User Data* parameter, and
    B. the abstract value “rejected (permanent)” as the D-START *Result* parameter value, and
III. Enter the *IDLE* state.

2.3.5.3.9 **DSC-start Service Request**

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

I. Create an AircraftPDUs APDU with a StartDownMessage~~ATCDownlinkMessage~~ APDU element containing:
    A. the abstract value “dsc” as the mode,
    B. if provided, the *CPDLC Message* parameter as the DownlinkMessage, or
    C. else, NULL as the DownlinkMessage,
II. Invoke D-START request with the following:
    A. the DSC-start service *ICAO Facility Designator* parameter value as the D-START *Called Peer ID* parameter value,
    B. the DSC-start service *Aircraft Identifier* parameter value as the D-START *Calling Peer ID* parameter value,
    C. Set the D-START *Quality of Service* parameters as follows:
        1. *Class of Communication* parameter from the DSC-start service request if provided,
        2. The abstract value of “high~~normal~~ priority flight safety messages”, as the D-START *QOS Priority* parameter value, and
        3. The abstract value of “low” as the D-START *QOS Residual Error Rate* parameter value, and
    D. the APDU as the D-START *User Data* parameter value;
III. Set DSC to the abstract value “true”,
IV. Start timer t_{start}, and
V. Enter the *START-REQ* state.

2.3.5.3.10 **CPDLC-message Service Request**

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

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,
b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and
c) Remain in the *DIALOGUE* state.

2.3.5.3.10.2 Upon receipt of a CPDLC-message service request, if the CPDLC-air-ASE is in the *END* state and DSC has the abstract value “false”, the CPDLC-air-ASE shall:

a) Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,
b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and
c) Remain in the *END* state.

2.3.5.3.11 **CPDLC-end Service Response**

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

I. Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,
II. Invoke D-END response with the following:
    A. if created, the APDU as the D-END *User Data* parameter value, and
    B. the abstract value "accepted", as the D-END *Result* parameter value, and
III. Enter the *IDLE* state.

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

I. Create an AircraftPDUs APDU with an ATCDownlinkMessage APDU-element based on the CPDLC-end service *CPDLC Message* parameter, if provided,
II. Invoke D-END response with the following:
    A. if created, the APDU as the D-END *User Data* parameter value, and
    B. the abstract value "rejected", as the D-END *Result* parameter value, and
III. Enter the *DIALOGUE* state

2.3.5.3.12 **DSC-end Service Request**

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

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

2.3.5.3.13 **CPDLC-user-abort Service Request**

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

I. Stop any timer,
II. If the CPDLC-user-abort service *Reason* parameter is provided, create an AircraftPDUs APDU with a CPDLCUserAbortReason APDU-element based on the CPDLC-user-abort service *Reason* parameter,
III. Else create an AircraftPDUs APDU with a CPDLCUserAbortReason [undefined~~cPDLC-user-abort~~] APDU-element
IV. Invoke D-ABORT request with the following:
    A. the D-ABORT *Originator* parameter set to the abstract value "user", and
    B. the APDU as the D-ABORT *User Data* parameter value, and
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

2.3.5.3.14 **D-ABORT Indication**

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

I. Stop any timer,

II. If the CPDLC-air-user is an active user, invoke CPDLC-user-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CPDLC-user-abort service *Reason* parameter value,
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

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

I. Stop any timer,
II. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided,
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

2.3.5.3.15 **D-P-ABORT Indication**

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

I. Stop any timer,
II. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-failure",
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

2.3.5.4 CPDLC-Air-ASE Exception Handling

2.3.5.4.1 A Timer Expires

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

I. Interrupt any current activity,
~~Stop any timer,~~
II. Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [timer-expired] APDU message element,
III. Invoke D-ABORT request with:
    A. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
    B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "timer-expired" as the CPDLC-provider abort service *Reason* parameter value,
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

2.3.5.4.2 Unrecoverable System Error

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

I. Stop any timer,
II. Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [undefined-error] APDU message element,
III. Invoke D-ABORT request with:
    A. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
    B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "undefined-error" as the CPDLC-provider abort service *Reason* parameter value,

V. If DSC has the abstract value “true”, set DSC to the abstract value “false”, and
VI. Enter the *IDLE* state.

2.3.5.4.3 Invalid PDU

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

I. Stop any timer,
II. Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [invalid-PDU] APDU message element,
III. Invoke D-ABORT request with:
    A. the abstract value “provider” as the D-ABORT *Originator* parameter value, and
    B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “invalid-PDU” as the CPDLC-provider abort service *Reason* parameter value,
V. If DSC has the abstract value “true”, set DSC to the abstract value “false”, and
VI. Enter the *IDLE* state.

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

a) Stop any timer,
b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value “invalid-PDU”-,
c) If DSC has the abstract value “true”, set DSC to the abstract value “false”, and
d) Enter the *IDLE* state.

2.3.5.4.4 Not Permitted PDU

2.3.5.4.4.1 If the *User Data* parameter of a D-START indication or D-DATA indication is a valid PDU, but is not a ~~permitted~~ PDU for which action is described within a given state ~~as defined~~ in 2.3.5.3, the CPDLC-air-ASE shall:

I. Stop any timer,
II. Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,
III. Invoke D-ABORT request with:
    A. the abstract value “provider” as the D-ABORT *Originator* parameter value, and
    B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value “not-permitted-PDU” as the CPDLC-provider abort service *Reason* parameter value ,
V. If DSC has the abstract value “true”, set DSC to the abstract value “false”, and
VI. Enter the *IDLE* state.

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

I. If the D-START *Result* parameter is set to the abstract value “accepted”, then
    A. Stop any timer,
    B. Create an AircraftPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,
    C. Invoke D-ABORT request with:
        1. the abstract value “provider” as the D-ABORT *Originator* parameter value, and
        2. the APDU as the D-ABORT *User Data* parameter value, and

II. If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider-abort service *Reason* parameter value,
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

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

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

a) Stop any timer,
b) If the CPDLC-air-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-error",
c) If DSC has the abstract value "true", set DSC to the abstract value "false", and
d) Enter the *IDLE* state.

2.3.5.5 CPDLC-Ground-ASE Protocol Description

2.3.5.5.1 Introduction

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

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

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

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

Note 1. — The states defined for the CPDLC-ground-ASE are the following.
a) IDLE
b) START-REQ,
c) START-IND,
d) DIALOGUE,
e) END, and
f) FORWARD.

Note 2. — The CPDLC-ground-user is an active user from:
I. the time it has invoked the CPDLC-start service request until:
A. receipt of a CPDLC-start service confirmation with Result parameter equal to the abstract value "rejected", or
B. receipt of a CPDLC-end service confirmation with the Result parameter equal to the abstract value "accepted", or
C. invocation of a CPDLC-user-abort service request, or
D. receipt of a CPDLC-user-abort service indication, or
E. receipt of a CPDLC-provider-abort service indication; or
II. the time it has received the CPDLC-start service indication until:
A. invocation of a CPDLC-start service response with Result parameter set to the abstract value "rejected", or
B. receipt of a CPDLC-end service confirmation with the Result parameter equal to the abstract value "accepted", or
C. invocation of a CPDLC-user-abort service request, or

D. receipt of CPDLC-user-abort service indication, or
E. receipt of a CPDLC-provider-abort service indication; or

III. the time it has received the DSC-start service indication until:
A. invocation of a DSC-start service response with Result parameter equal to the abstract value "rejected", or
B. invocation of a CPDLC-user-abort service request, or
C. receipt of CPDLC-user-abort service indication, or
D. receipt of a CPDLC-provider-abort service indication; or

IV. the time it has invoked the CPDLC-forward service request until:
A. receipt of a CPDLC-forward service confirmation,
B. invocation of a CPDLC-user-abort service request, or
C. receipt of CPDLC-user-abort service indication, or
D. receipt of a CPDLC-provider-abort service indication.

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

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

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

2.3.5.5.2 **D-START Indication**

2.3.5.5.2.1 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a 24 bit Aircraft Address and the D-START *User Data* parameter contains an AircraftPDUs [StartDownMessage] APDU with the APDU-element mode "cpdlc", the CPDLC-ground-ASE shall:

I. Invoke CPDLC-start service indication containing the following:
A. the D-START *Calling Peer ID* parameter value as the CPDLC-start service *Calling Peer Identifier* parameter value,
B. the D-START *QOS Routing Class* parameter value as the CPDLC-start service *Class of Communication* parameter value,
C. if the AircraftPDUs APDU-element contained in the D-START *User Data* parameter is an ATCDownlinkMessage, set the AircraftPDUs APDU-element as the CPDLC-start service *CPDLC Message* parameter value, and
II. Enter the *START-IND* state.

2.3.5.5.2.2 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is a 24 bit Aircraft Address and the D-START *User Data* parameter contains an AircraftPDUs [StartDownMessage]APDU with the APDU-element mode "dsc", the CPDLC-ground-ASE shall:

I. Invoke DSC-start service indication containing the following:
A. the D-START *Calling Peer ID* parameter value as the DSC-start service *Aircraft Identifier* parameter value,
B. the D-START *QOS Routing Class* parameter value as the CPDLC-start service *Class of Communication* parameter value,
C. 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,
II. Set DSC to "true", and
III. Enter the *START-IND* state.

2.3.5.5.2.3 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is an 8 character ICAO Facility Designator and the D-START *User Data* parameter contains a GroundPDUs [ATCForwardMessage] APDU and the CPDLC-ground-ASE supports the CPDLC-forward service, the CPDLC-ground-ASE shall:

I. If the D-START *DS User Version Number* parameter value is equal to the CPDLC-ground-ASE version number:
   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 D-START GroundPDUs APDU-element as the CPDLC-forward service *CPDLC Message* parameter value, and
   B. Create a GroundPDUs APDU with an ATCForwardResponse [success] APDU element,
   C. Invoke D-START response with the following:
      1. the APDU as the D-START User Data parameter value, and
      2. the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and
   D. Remain in the *IDLE* state.

II. If the D-START *DS User Version Number* parameter value is not equal to the CPDLC-ground-ASE version number:
   A. Create a GroundPDUs APDU with an ATCForwardResponse [version-not-equal~~incompatible-version~~] APDU element,
   B. Invoke D-START response with the following:
      1. the CPDLC-ground-ASE version number as the D-START *DS User Version Number* parameter value,
      2. the APDU as the D-START *User Data* parameter value, and
      3. the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and
   C. Remain in the *IDLE* state.

2.3.5.5.2.4 Upon receipt of a D-START indication, if the CPDLC-ground-ASE is in the *IDLE* state, and the abstract value of the D-START *Calling Peer ID* parameter is an 8 character ICAO Facility Designator and the D-START *User Data* parameter contains a GroundPDUs APDU and the APDU element is an ATCForwardMessage and the CPDLC-ground-ASE does not support the CPDLC-forward service, the CPDLC-ground-ASE shall:

I. Create a GroundPDUs APDU with an ATCForwardResponse [service-not-supported] APDU element,
II. Invoke D-START response with the following:
   A. the APDU as the D-START *User Data* parameter value, and
   B. the abstract value "rejected (permanent)" as the D-START *Result* parameter value, and
III. Remain in the *IDLE* state.

2.3.5.5.3 **D-START Confirmation**

2.3.5.5.3.1 Upon receipt of a D-START confirmation, if the CPDLC-ground-ASE is in the *START-REQ* state and if the D-START *Result* parameter has the abstract value "accepted", and DSC has the abstract value of "false" and D-START *User Data* parameter is not provided, the CPDLC-ground-ASE shall:

I. Stop timer t_{start},
II. Invoke CPDLC-start service confirmation containing the abstract value "accepted" as the CPDLC-start service *Result* parameter value, and
III. Enter the *DIALOGUE* state.

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

I. Stop timer t_{start},
II. Invoke CPDLC-start service confirmation containing the following:
   A. if the D-START *User Data* parameter is provided, the APDU contained in the D-START *User Data* parameter as the CPDLC-start service *Reject Reason* parameter value, and
   B. the abstract value "rejected" as the CPDLC-start service *Result* parameter value, and
III. Enter the *IDLE* state.

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

I. If the D-START *DS User Version Number* parameter value is equal to the CPDLC-ground-ASE version number:
   A. Stop timer t_{start},
   B. Invoke CPDLC-forward service confirmation with the D-START GroundPDUs APDU-element as the CPDLC-forward service *Result* parameter value, and
   C. Enter the *IDLE* state.

II. If the D-START *DS User Version Number* parameter value is not equal to the CPDLC-ground-ASE version number:
   A. Stop timer t_{start},
   B. Invoke CPDLC-forward service confirmation with the following:
      1. the D-START GroundPDUs APDU-element as the CPDLC-forward service *Result* parameter value,
      2. the D-START *DS User Version Number* parameter value as the CPDLC-forward service *ASE Version Number* parameter value, and
   C. Enter the *IDLE* state.

2.3.5.5.4 **D-DATA Indication**

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

I. 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
II. Remain in the *DIALOGUE* state.

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

I. 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
II. Remain in the *END* state.

2.3.5.5.5 **D-END Indication**

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

I. 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
II. Enter the *END* state

2.3.5.5.6 **D-END Confirmation**

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

I. Invoke CPDLC-end service confirmation with:
   A. The APDU contained in the D-END *User Data* parameter as the CPDLC-end service *CPDLC Message* parameter value, if provided, and

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

II. Enter the *IDLE* state.

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

I. Invoke CPDLC-end service confirmation with:
   A. The APDU contained in the D-END *User Data* parameter as the CPDLC-end service *CPDLC Message* parameter value, if provided, and
   B. The abstract value "rejected~~accepted~~" as the CPDLC-end service *Result* parameter value, and

II. Enter the *DIALOGUE* state.

2.3.5.5.7 **CPDLC-start Service Request**

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

I. Create a GroundPDUs APDU with an UplinkMessage APDU element containing:
   A. if provided, the *CPDLC Message* parameter as the UplinkMessage, or
   B. else, NULL as the UplinkMessage,

II. Invoke D-START request with the following:
   A. the CPDLC-start service *Called Peer Identifier* parameter value as the D-START *Called Peer ID* parameter value,
   B. the CPDLC-start service *Calling Peer Identifier* parameter value as the D-START *Calling Peer ID* parameter value,
   C. the D-START *Quality of Service* parameters set as follows:
      1. if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START *QOS Routing Class* parameter value, or
      2. The abstract value of "high~~normal~~ priority flight safety messages", as the D-START *QOS Priority* parameter value, and
      3. The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and
   D. the APDU as the D-START *User Data* parameter value;

III. Start timer t_{start}, and

IV. Enter the *START-REQ* state.

2.3.5.5.8 **CPDLC-start Service Response**

2.3.5.5.8.1 Upon receipt of a CPDLC-start service response, if the CPDLC-ground-ASE is in the *START-IND* state and the CPDLC-start service *Result* parameter has the abstract value "accepted" and the CPDLC-start service *Reject Reason* parameter is not provided, and DSC has the abstract value "false", the CPDLC-ground-ASE shall:

I. Invoke D-START response with the abstract value "accepted" as the D-START *Result* parameter value, and

II. Enter the *DIALOGUE* state.

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

I. If the CPDLC-start service *Reject Reason* parameter is provided, create a GroundPDUs APDU with an ATCUplinkMessage APDU element based on the *Reject Reason* parameter,

II. Invoke D-START response with the following:
   A. The APDU as the D-START *User Data* parameter value; if the *Reject Reason* parameter was provided, and
   B. the abstract value "rejected (permanent)"as the D-START *Result* parameter value, and

III. Enter the *IDLE* state.

2.3.5.5.9 **DSC-start Service Response**

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

I. Invoke D-START response with the abstract value “accepted” as the D-START *Result* parameter value, and
II. Enter the *DIALOGUE* state.

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

I. 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,
II. Invoke D-START response with the following:
    A. The APDU element as D-START *User Data* parameter value; if the *Reject Reason* parameter was provided, and
    B. the abstract value “rejected (permanent)”as the D-START *Result* parameter value,
III. Set DSC to the abstract value “false”, and
IV. Enter the *IDLE* state.

2.3.5.5.10 **CPDLC-message Service Request**

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

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,
b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and
c) Remain in the *DIALOGUE* state.

2.3.5.5.10.2 Upon receipt of a CPDLC-message service request, if the CPDLC-ground-ASE is in the *END* state and DSC has the abstract value “true”, the CPDLC-ground-ASE shall:

a) Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the CPDLC-message service *CPDLC Message* parameter,
b) Invoke D-DATA request with the APDU as the D-DATA *User Data* parameter value, and
c) Remain in the *END* state.

2.3.5.5.11 **CPDLC-end Service Request**

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

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

2.3.5.5.12 **DSC-end Service Response**

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

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

II. Invoke D-END response with the following:
  A. the APDU as the D-END *User Data* parameter; if provided, and
  B. the abstract value "accepted" as the D-END *Result* parameter value,
III. Set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

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

I. Create a GroundPDUs APDU with an ATCUplinkMessage APDU-element based on the DSC-end service *CPDLC Message* parameter , if provided,
II. Invoke D-END response with the following:
  A. the APDU as the D-END *User Data* parameter; if provided, and
  B. the abstract value "rejected" as the D-END *Result* parameter value, and
III. Enter the *DIALOGUE* state.

2.3.5.5.13 **CPDLC-forward Service Request**

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

I. Create a GroundPDUs APDU with an ATCForwardMessage APDU element based on the CPDLC-forward service *CPDLC Message* parameter,
II. Invoke D-START request with the following:
  A. the CPDLC-forward service *Called ICAO Facility Designator* parameter value as the D-START *Called Peer ID* parameter value,
  B. the CPDLC-start service *Calling ICAO Facility Designator* parameter value as the D-START *Calling Peer ID* parameter value,
  C. the D-START *Quality of Service* parameters set as follows:
    1. if provided, the CPDLC-start service *Class of Communication* parameter value as the D-START *QOS Routing Class*, or
    2. The abstract value of "high~~normal~~ priority flight safety messages", as the D-START *QOS Priority* parameter value, and
    3. The abstract value of "low" as the D-START *QOS Residual Error Rate* parameter value, and
  D. the APDU as the D-START *User Data* parameter value;
III. Start timer t_{start}, and
IV. Enter the *FORWARD* state.

2.3.5.5.14 **CPDLC-user-abort Service Request**

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

I. Stop any timer,
II. If the CPDLC-user-abort service *Reason* parameter is provided, create a GroundPDUs APDU with a CPDLCUserAbortReason APDU-element based on the CPDLC-user-abort service *Reason* parameter,
III. Else create a GroundPDUs APDU with a CPDLCUserAbortReason [undefined~~cPDLC-user-abort~~] APDU-element
IV. Invoke D-ABORT request with the following:
  A. the D-ABORT *Originator* parameter set to the abstract value "user", and
  B. the APDU as the D-ABORT *User Data* parameter value, and
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

2.3.5.5.15 **D-ABORT Indication**

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

I. Stop any timer,
II. If the CPDLC-ground-user is an active user, invoke CPDLC-user-abort service indication with the APDU contained in the D-ABORT *User Data* parameter as the CPDLC-user-abort service *Reason* parameter value,
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

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

I. Stop any timer,
II. If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the D-ABORT *User Data* parameter as the CPDLC-provider-abort service *Reason* parameter value, if provided,
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

2.3.5.5.16 **D-P-ABORT Indication**

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

I. Stop any timer,
II. If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-failure",
III. If DSC has the abstract value "true", set DSC to the abstract value "false", and
IV. Enter the *IDLE* state.

2.3.5.6 CPDLC-Ground-ASE Exception Handling

2.3.5.6.1 A Timer Expires

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

I. Interrupt any current activity,
~~Stop any timer,~~
II. Create a GroundPDUs APDU with a CPDLCProviderAbortReason [timer-expired] APDU message element,
III. Invoke D-ABORT request with:
   A. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
   B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "timer-expired" as the CPDLC-provider abort service *Reason* parameter value",
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

2.3.5.6.2 Unrecoverable System Error

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

I. Stop any timer,
II. Create a GroundPDUs APDU with a CPDLCProviderAbortReason [undefined-error] APDU message element,

III. Invoke D-ABORT request with:
  A. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
  B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "undefined-error" as the CPDLC-provider abort service *Reason* parameter value",
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

2.3.5.6.3 Invalid PDU

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

I. Stop any timer,
II. Create a GroundPDUs APDU with a CPDLCProviderAbortReason [invalid-PDU] APDU message element,
III. Invoke D-ABORT request with:
  A. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
  B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "invalid-PDU" as the CPDLC-provider abort service *Reason* parameter value,
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

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

a) Stop any timer,
b) If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "invalid-PDU",
c) If DSC has the abstract value "true", set DSC to the abstract value "false", and
d) Enter the *IDLE* state.

2.3.5.6.4 Not Permitted PDU

2.3.5.6.4.1 If the *User Data* parameter of a D-START indication or D-DATA indication is a valid PDU, but is not a permitted PDU for which action is described within a given state as defined in 2.3.5.5~~3~~, the CPDLC-ground-ASE shall:

I. Stop any timer,
II. Create a GroundPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,
III. Invoke D-ABORT request with:
  A. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
  B. the APDU as the D-ABORT *User Data* parameter value, and
IV. If the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider abort service *Reason* parameter value",
V. If DSC has the abstract value "true", set DSC to the abstract value "false", and
VI. Enter the *IDLE* state.

2.3.5.6.4.2 If the *User Data* parameter of a D-START confirmation is a valid PDU, but is not a permitted PDU for which action is described within a given state as defined in 2.3.5.5~~3~~, the CPDLC-ground-ASE shall:

I. Stop any timer,
II. If the D-START *Result* parameter is set to the abstract value "accepted", then

A. Create a GroundPDUs APDU with a CPDLCProviderAbortReason [not-permitted-PDU] APDU message element,
B. Invoke D-ABORT request with:
1. the abstract value "provider" as the D-ABORT *Originator* parameter value, and
2. the APDU as the D-ABORT *User Data* parameter value, and

III. If the CPDLC-user is an active user, invoke CPDLC-provider-abort service indication with the abstract value "not-permitted-PDU" as the CPDLC-provider-abort service *Reason* parameter value,
IV. If DSC has the abstract value "true", set DSC to the abstract value "false", and
V. Enter the *IDLE* state.

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

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

a) Stop any timer,
b) if the CPDLC-ground-user is an active user, invoke CPDLC-provider-abort service indication with the CPDLC-provider-abort service *Reason* parameter set to the abstract value "communication-service-error",
c) If DSC has the abstract value "true", set DSC to the abstract value "false", and
d) Enter the *IDLE* state.

2.3.5.7 CPDLC ASE State Tables

2.3.5.7.1 Priority

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

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

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

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

STATE ⇒ EVENT ⇓	**IDLE**	**START-REQ**	**START-IND**	**DIALOGUE**	**END**
Dialogue Service Events					
D-START Indication APDU = UplinkMessage	• CPDLC-start indication ⇒START-IND	cannot occur	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "false" No *User Data*	cannot occur	• Stop timer t_{start} • CPDLC-start confirmation ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user", DSC="false" if *User Data*, APDU = ATCUplinkMessage	cannot occur	• Stop timer t_{start} • CPDLC-start confirmation ⇒*IDLE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "accepted", DSC = "true" No *User Data*	cannot occur	• Stop timer t_{start} • DSC-start confirmation, ⇒*DIALOGUE*	cannot occur	cannot occur	cannot occur
D-START Confirmation *Result* "rejected (permanent)" and *Reject Source* "DS user", DSC= "true" if *User Data*, APDU = ATCUplinkMessage	cannot occur	• Stop timer t_{start} • DSC-start confirmation • Set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur
D-DATA Indication APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	• CPDLC-message indication ⇒*DIALOGUE*	• if DSC="true" • CPDLC-message indication ⇒*END* • else not permitted
D-END Indication: DSC="false" if *User Data*, APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	• CPDLC-end indication ⇒*END*	cannot occur

D-END Confirmation: DSC="true" *Result* "accepted" if *User Data*, APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	cannot occur	• DSC-end confirmation • Set DSC "false" ⇒*IDLE*
D-END Confirmation: DSC="true", *Result* "rejected" if *User Data*, APDU = ATCUplinkMessage	cannot occur	cannot occur	cannot occur	cannot occur	• DSC-end confirmation ⇒*DIALOGUE*
CPDLC-User Events	•				
CPDLC-start Request	• D-START request • Start timer t_{start} ⇒*START-REQ*	not permitted	not permitted	not permitted	not permitted
CPDLC-start Response DSC="false" *Result* "accepted"	not permitted	not permitted	• D-START response ⇒*DIALOGUE*	not permitted	not permitted
CPDLC-start Response DSC="false" *Result* "rejected"	not permitted	not permitted	• D-START response ⇒*IDLE*	not permitted	not permitted
DSC-start Request	• D-START request • set DSC ="true" • Start timer t_{start} ⇒*START-REQ*	not permitted	not permitted	not permitted	not permitted
CPDLC-message Request	not permitted	not permitted	not permitted	• D-DATA request ⇒*DIALOGUE*	• if DSC="false" • D-DATA request ⇒*END* • else not permitted
CPDLC-end Service Response DSC = "false" *Result* "accepted"	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*IDLE*
CPDLC-end Service Response DSC = "false" *Result* "rejected"	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*DIALOGUE*

DSC-end Request: DSC = "true"	not permitted	not permitted	not permitted	• D-END request ⇒*END*	not permitted
ABORT Events					
CPDLC-user-abort Request	not permitted	• Stop timer t_{start}, if set • D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer t_{start}, if set • D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*	• D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*	• D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*
D-ABORT Indication *Originator* "user" ~~APDU = CPDLCAbortReason~~	cannot occur	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*
D-ABORT Indication *Originator* "provider"	cannot occur	• Stop timer T_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC "false" ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC "false" ⇒*IDLE*
D-P-ABORT indication ~~DSC = "true"~~	cannot occur	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*
T_{start} Expires	cannot occur	• D-ABORT request • CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur

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

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

D-DATA Indication	cannot occur	cannot occur	cannot occur	• CPDLC-message indication ⇒*DIALOGUE*	• <u>if DSC="false"</u> • CPDLC-message indication ⇒*END* ⇒<u>else not permitted</u>	cannot occur
D-END Indication: DSC="true"	cannot occur	cannot occur	cannot occur	• DSC-end indication ⇒*END*	cannot occur	cannot occur
D-END Confirmation: DSC = "false" *Result* "accepted"	cannot occur	cannot occur	cannot occur	cannot occur	• CPDLC-end confirmation ⇒*IDLE*	cannot occur
D-END Confirmation: DSC = "false" *Result* "rejected"	cannot occur	cannot occur	cannot occur	cannot occur	• CPDLC-end confirmation ⇒*DIALOGUE*	cannot occur
CPDLC-User Events						
CPDLC-start Request	• D-START request • Start timer t_{start} ⇒*START-REQ*	not permitted	not permitted	not permitted	not permitted	not permitted
CPDLC-start Response DSC = "false" *Result* "accepted"	not permitted	not permitted	• D-START response ⇒*DIALOGUE*	not permitted	not permitted	not permitted
CPDLC-start Response DSC = "false" *Result* "rejected"	not permitted	not permitted	• D-START response ⇒*IDLE*	not permitted	not permitted	not permitted
DSC-start Response DSC = "true" *Result* "accepted"	not permitted	not permitted	• D-START response ⇒*DIALOGUE*	not permitted	not permitted	not permitted
DSC-start Response DSC = "true" *Result* "rejected"	not permitted	not permitted	• D-START response • <u>Set DSC= "false"</u> ⇒*IDLE*	not permitted	not permitted	not permitted

CPDLC-message Request	not permitted	not permitted	not permitted	• D-DATA request ⇒*DIALOGUE*	• if DSC="true" • D-DATA request ⇒*END* • else not permitted	not permitted
CPDLC-end Request: DSC = "false"	not permitted	not permitted	not permitted	• D-END request ⇒*END*	not permitted	not permitted
DSC-end Service Response DSC = "true" *Result* "accepted"	cannot occur	cannot occur	cannot occur	not permitted	• D-END response • Set DSC = "false" ⇒*IDLE*	not permitted
DSC-end Service Response DSC = "true" *Result* "rejected"	cannot occur	cannot occur	cannot occur	not permitted	• D-END response ⇒*DIALOGUE*	not permitted
CPDLC-forward Request	• D-START request • Start timer t_{start} ⇒*FORWARD*	not permitted	not permitted	not permitted	not permitted	not permitted
ABORT Events						
CPDLC-user-abort Request	not permitted	• Stop timer t_{start}, if set • D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer t_{start}, if set • D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*	• D-ABORT request • If DSC = "true", set DSC = "false" ⇒*IDLE*	• D-ABORT request ⇒*IDLE* ⇒If DSC = "true", set DSC = "false"	• Stop timer t_{start}; if set • D-ABORT request ⇒*IDLE*
D-ABORT Indication *Originator* "provider"	cannot occur	• Stop timer T_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-provider-abort indication ⇒*IDLE*

D-ABORT Indication *Originator* "user"	cannot occur	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-user-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer T_{start}, if set • If active user: CPDLC-user-abort indication ⇒*IDLE*
D-P-ABORT indication	cannot occur	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• If active user: CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	• Stop timer t_{start}, if set • If active user: CPDLC-provider-abort indication ⇒*IDLE*
T_{start} Expires	cannot occur	• D-ABORT request • CPDLC-provider-abort indication • If DSC = "true", set DSC = "false" ⇒*IDLE*	cannot occur	cannot occur	cannot occur	• D-ABORT request • CPDLC-provider-abort indication ⇒*IDLE*

2.3.6. COMMUNICATION REQUIREMENTS

2.3.6.1 Encoding Rules

2.3.6.1.1 The CPDLC application shall use PER as defined in ISO/IEC 8825-2, using the Basic Unaligned variant to encode/decode the ASN.1 message structure and content specified in 2.3.4~~, or a functionally equivalent means which provides the same result~~.

2.3.6.2 Dialogue Service Requirements

2.3.6.2.1 Primitive Requirements

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

2.3.6.2.2 Quality-of-Service Requirements

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

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

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

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

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

Note. — ATSC values are defined in 1.2.

~~**CPDLC AE Control Functions Requirements**~~

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

2.3.7. CPDLC USER REQUIREMENTS

2.3.7.1 General

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

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

Note 3. — A CPDLC message is:

a) what the CPDLC-user provides the CPDLC-service as the CPDLC Message or Reject Reason parameter, when invoking a CPDLC service request or response primitive, or
b) what the CPDLC-user receives in the same parameters from the CPDLC service indication or confirmation primitives.

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

2.3.7.1.1 General CPDLC Service Requirements

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

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

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

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

2.3.7.2 CPDLC Message Generation Requirements

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

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

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

2.3.7.2.1 Message Composition

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

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

2.3.7.2.1.3 For ground/ground messages, the message header shall be composed of a time stamp, the aircraft flight identification, and the airframe identification to which the message refers.

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

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

a) a message identification number,
b) a message reference number only if the message is a response message,
c) date and time,
d) a logical acknowledgment indication, if required,
e) from one to five message element identifiers, and
f) data as required for each message element identification included.

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

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

2.3.7.2.2 Message Identification Number

Note 1. — A message identification number pertains to a single peer to peer dialogue.

Note 2. — Message identification numbers used by a CPDLC ground system for uplink messages to an aircraft have no relationship to the message identification numbers used by the same ground system another aircraft.

Note 3. — Similarly, message identification numbers used by a CPDLC aircraft for downlink messages to a CPDLC ground system have no relationship to the message identification numbers used by the same aircraft with another ground system.

Note 4. — There is no relationship between message identification numbers assigned and managed by a the CPDLC ground system and those message identification numbers assigned and managed by the aircraft.

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

2.3.7.2.2.2 A message identification number shall be deemed currently in use until:

a) if the message does not allow a response, the message is sent, or
b) if the message requires a response, the closure response is received.

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

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

2.3.7.2.3.1 The ERROR [error information] message element (uplink message element 159, and downlink message 62) shall be sent only as a single message element CPDLC message.

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

2.3.7.2.3.3 The NEXT DATA AUTHORITY [~~icao~~ facility designation~~or~~] message element (uplink message element 160) shall be sent only as a single message element CPDLC message.

2.3.7.3 CPDLC Message Receipt Requirements

2.3.7.3.1 Message Attributes

Note 1. — Message attributes dictate certain message handling requirements for the CPDLC-user receiving a message. Each CPDLC message has Urgency, Alert, and Response attributes.

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

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

2.3.7.3.1.2 When a message contains multiple message elements, the highest precedence message element attribute for each attribute type associated with any element in the message shall be the message attribute for each attribute type for the entire message.

2.3.7.3.2 Urgency Requirements

Note 1. — The Urgency (URG) attribute delineates the queuing requirements for received messages that are displayed to the end-user. The same Urgency attribute types are used for both air/ground and ground/ground messages.

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

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

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

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

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

2.3.7.3.3 Alerting Requirements

Note 1. — The alert (ALRT) attribute delineates the type of end-user alerting required by the CPDLC-user upon message receipt. The same Alert attribute types are used for both air/ground and ground/ground messages.

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

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

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

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

2.3.7.3.4 Response Attribute

Note ~~1~~. — The response (RESP) attribute mandates CPDLC-user response requirements for a given message element.~~:~~ Response message attribute only apply to air/ground messages.

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

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

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

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

Table 2.3.7~~1~~-4. Response Attribute (Down-Link)

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

2.3.7.3.5 CPDLC/DSC Distinction

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

2.3.7.3.6 Air/Ground - Ground/Ground Distinction

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

2.3.7.3.7 Logical Acknowledgment Prohibited

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

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

2.3.7.3.8 Message Reference Numbers

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

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

2.3.7.3.9 Message Response Requirements

2.3.7.3.9.1 **Recommendation.** ~~Note 1.—~~ *A message sequence initiated by data link should be closed by data link.*

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

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

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

2.3.7.3.9.5 If the CPDLC-air user has not issued a DSC-end service request primitive, or if the CPDLC-air-user has issued a DSC-end service request primitive for which a DSC-end service confirmation primitive has been received with the *Result* parameter containing the abstract value “rejected” then, if a message is received that requires a response, the CPDLC-air-user shall either:

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

2.3.7.3.9.6 If the CPDLC-ground-user has not issued a CPDLC-end service request primitive, or if the CPDLC-ground-user has issued a CPDLC-end service request primitive for which a CPDLC-end service confirmation primitive has been received with the *Result* parameter containing the abstract value "rejected" then, if a message is received that requires a response, the CPDLC-air-user shall either:

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

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

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

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

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

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

2.3.7.3.9.12 When the CPDLC-air-user receives a message with 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.

2.3.7.3.9.13 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.

2.3.7.3.9.14 When the CPDLC-air-user receives a message with a R RESP attribute, the only permitted responses shall be messages that contain a LOGICAL ACKNOWLEDGMENT (if required), STANDBY, ROGER, or ERROR message element.

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

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

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

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

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

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

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

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

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

2.3.7.3.10 Error Conditions

2.3.7.3.10.1 Duplicate Message Identification Numbers

2.3.7.3.10.1.1 If a CPDLC message is received containing an identification number identical to that of an identification number currently in use the CPDLC-user shall invoke the CPDLC-user-abort request service with a CPDLC message containing the CPDLC<u>User</u>AbortReason with value [duplicate-message-identification-number] as the *Reason* parameter.

2.3.7.3.10.2 Invalid Reference Number

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

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

2.3.7.3.10.3 No Available Message Identification Numbers

2.3.7.3.10.3.1 If the CPDLC-user attempts to send a CPDLC message and all message identification numbers are currently in use, the CPDLC-user shall invoke the CPDLC-user-abort request with <u>a</u> CPDLC ~~a~~ message containing the CPDLC<u>User</u>AbortReason with the value [no-message-identification-numbers-available] as the *Reason* parameter.

2.3.7.3.10.4 Insufficient <u>Resources</u>~~Storage~~

2.3.7.3.10.4.1 If the CPD<u>L</u>C-user receives a message and has insufficient <u>resources</u>~~storage~~ to handle the message, the CPDLC-user shall ~~the CPDLC-user shall~~:

a) invoke CPDLC-message request with the ERROR [errorinformation] message element with the [insufficient<u>Resources</u>~~MessageStorageCapacity~~] value as the *CPDLC Message* parameter, and
b) disregard the received message.

2.3.7.3.10.5 ERROR Message Response

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

2.3.7.3.10.5.2 If an ERROR message is sent in response to a CPDLC message the received CPDLC message shall be disregarded ~~and no further action shall be taken on that message~~.

2.3.7.4 CPDLC-air-user Requirements

2.3.7.4.1 The CPDLC-start Service

2.3.7.4.1.1 Invoking the CPDLC-start request

2.3.7.4.1.1.1 If there is no CPDLC service, the only CPDLC service primitives the CPDLC-air-user shall be permitted to invoke are the CPDLC-start request or the DSC-start request.

2.3.7.4.1.1.2 The CPDLC-air-user shall only be permitted to invoke the CPDLC-start request with:

a) any ground system, if there is no existing CPDLC service for the CPDLC-air-user, or
b) the Next Data Authority, if the CPDLC-air-user has received a message from the Current Data Authority designating a Next Data Authority.

2.3.7.4.1.1.3 If a CPDLC-air-user has invoked a CPDLC-start request, the CPDLC-air-user shall be prohibited from invoking any CPDLC-service primitive 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.

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

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

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

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

2.3.7.4.1.2.3 If a CPDLC-start indication is received from either the Current Data Authority or the Next Data Authority, and this results in a second CPDLC dialogue being established with a given ground system, the CPDLC-air-user shall invoke the CPDLC-user-abort request primitive for the first connection with that ground system.

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

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

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

2.3.7.4.1.2.7 If the CPDLC-air-user sets the CPDLC-response *Result* parameter to the abstract value "accepted" the CPDLC-air-user shall:

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

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

2.3.7.4.1.3 Receipt of a CPDLC-start confirmation

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

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

2.3.7.4.2 The DSC-start Service

2.3.7.4.2.1 Invoking the DSC-start request

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

2.3.7.4.2.1.2 A CPDLC-air-user shall only be permitted to invoke the DSC-start-service request primitive if the CPDLC-air-user has no existing DSC dialogue.

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

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

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

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

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

2.3.7.4.2.3 Receipt of a DSC-start confirmation

2.3.7.4.2.3.1 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 designationor ~~or~~ contained in DSC-start request *ICAO Facility Designator* parameter,
b) Associate the CPDLC-ASE invocation with a Downstream Data Authority.

2.3.7.4.3 The CPDLC-message Service

2.3.7.4.3.1 Receipt of a CPDLC-message Indication

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

2.3.7.4.3.1.2 If a CPDLC-message indication is received from the Current Data Authority containing 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 designationor ~~or~~ 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 CPDLCUserAbortReason value [no-longer-next-data-authority]; and
c) Then designate the ground system indicated in the CPDLC message from the CPDLC-message indication as the Next Data Authority.

2.3.7.4.3.1.3 If a CPDLC-message indication is received from the Current Data Authority containing 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.

2.3.7.4.3.1.4 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.

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

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

2.3.7.4.4 The CPDLC-end Service

2.3.7.4.4.1 The CPDLC-end Service Request

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

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

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

a) Invoke CPDLC-end response with
    1) the *CPDLC Message* parameter containing a message with the message element NOT CURRENT DATA AUTHORITY, and
    2) the *Result* parameter set to the abstract value “rejected”, and
b) Disregard any message provided in the CPDLC-end indication *CPDLC Message* parameter.

2.3.7.4.4.2.2 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.

2.3.7.4.4.2.3 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.

2.3.7.4.4.2.4 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”.

2.3.7.4.4.2.5 If a CPDLC-end indication is received that is from the Current Data Authority without a *CPDLC Message* parameter or with a *CPDLC Message* parameter containing a message with the response attribute N and not requiring a logical acknowledgment, 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".

2.3.7.4.4.2.6 Upon invoking CPDLC-end response with *Result* parameter set to "accepted", the CPDLC-air-user shall:

a) delete any association with a ground system and Current Data Authority, and
b) if a ground system is designated as Next Data Authority and an association with a CPDLC-ASE exists, replace the Next Data Authority association with a Current Data Authority association, or
c) if a ground system is designated as Next Data Authority and no association with a CPDLC-ASE exists, delete Next Data Authority association with any ground system.

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

2.3.7.4.5 The DSC-end Service

2.3.7.4.5.1 The DSC-end Request

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

2.3.7.4.5.1.2 If a CPDLC-air user has invoked a DSC-end service request primitive, the air-user shall be prohibited from invoking a any CPDLC service primitive with this ground system (except the CPDLC-user-abort request primitive) until it receives a DSC-end service confirmation primitive.

2.3.7.4.6 The CPDLC-user-abort Service

2.3.7.4.6.1 Issuing a CPDLC-user-abort Request

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

2.3.7.4.6.2 Receipt of a CPDLC-abort Indication

2.3.7.4.6.2.1 If the CPDLC-air-user receives a CPDLC-user-abort indication from the Current Data Authority or a CPDLC-provider-abort indication that causes the ASE invocation associated with the Current Data Authority to cease to exist, the CPDLC-air-user shall:

a) Delete any association of a ground system to a Current Data Authority,
b) If a ground system is designated as Next Data Authority and an association with a CPDLC-ASE exists, invoke CPDLC-user-abort request with the *Reason* parameter set to the value [current-data authority-abort], and
c) Delete any association of a ground system to a Next Data Authority.

2.3.7.4.6.2.2 If the CPDLC-air-user receives a CPDLC-user-abort indication from the Next Data Authority or receives a CPDLC-provider-abort indication that causes the ASE invocation associated with the Next Data Authority to cease to exist, the CPDLC-air-user shall continue to maintain the association of the ground system to the Next Data Authority.

2.3.7.5 CPDLC-Ground-User Requirements

2.3.7.5.1 The CPDLC-start Service

2.3.7.5.1.1 Invoking the CPDLC-start request

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

2.3.7.5.1.1.2 If a CPDLC-ground-user has invoked a CPDLC-start request, the CPDLC-ground-user shall be prohibited from invoking any CPDLC-service primitive, except the CPDLC-user-abort request with that aircraft, until after it has received a CPDLC-start confirmation.

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

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

a) invoke the CPDLC-start response with the Result parameter set to the abstract value “accepted”, and
b) invoke the CPDLC-user-abort request for the first CPDLC dialogue with that aircraft.

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

2.3.7.5.1.2.3 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value “rejected” then the *Reject Reason* shall be an uplink message containing only the SERVICE UNAVAILABLE message element.

2.3.7.5.1.2.4 If the CPDLC-ground-user sets the CPDLC-start response *Result* parameter to the abstract value “rejected” the CPDLC-ground-user shall disregard any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter.

2.3.7.5.1.2.5 If the CPDLC-ground-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” any CPDLC message contained in the CPDLC-start indication *CPDLC Message* parameter shall be processed.

2.3.7.5.1.2.6 If the CPDLC-ground-user sets the CPDLC-response *Result* parameter to the abstract value “accepted” the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in CPDLC-start indication *Calling Peer Identifier* parameter.

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

2.3.7.5.1.3 Receipt of a CPDLC-start confirmation

2.3.7.5.1.3.1 If a CPDLC-start confirmation has been received with a *Result* parameter containing the abstract value “accepted” the CPDLC-ground-user shall establish an association between a CPDLC-ASE invocation and a 24 bit aircraft address contained in CPDLC-start request *Called Peer Identifier* parameter.

2.3.7.5.2 The DSC-start Service

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

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

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

a) invoke the DSC-start response with the *Result* parameter set to the abstract value “accepted”, and
b) invoke the CPDLC-user-abort request for the first DSC dialogue with that aircraft.

2.3.7.5.2.1.3 If the CPDLC-ground-user sets the DSC-start response *Result* parameter to the abstract value “rejected” then the *Reject Reason* shall be an uplink message containing only the SERVICE UNAVAILABLE message element.

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

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

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

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

2.3.7.5.3 The CPDLC-end Service

2.3.7.5.3.1 The CPDLC-end Request

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

2.3.7.5.3.1.2 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.

2.3.7.5.3.1.3 If a CPDLC-ground user has invoked a CPDLC-end service request primitive, the ground-user shall be prohibited from invoking any CPDLC service primitive with this aircraft, except the CPDLC-user-abort request primitive, until after It has received a CPDLC-end service confirmation primitive.

2.3.7.5.4 The DSC-end Service

2.3.7.5.4.1 Receipt of a DSC-end Indication and Invoking DSC-end Response

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

2.3.7.5.4.1.2 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.

2.3.7.5.4.1.3 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.

2.3.7.5.4.1.4 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”.

2.3.7.5.4.1.5 If a DSC-end indication is received without a *CPDLC Message* parameter, or with a *CPDLC Message* parameter containing a message with the response attribute N and not requiring a logical acknowledgment, 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”.

2.3.7.5.5 The CPDLC-forward Service

2.3.7.5.5.1 Invoking the CPDLC-forward request

2.3.7.5.5.1.1 Only the CPDLC-ground-user shall be permitted to invoke the CPDLC-forward request.

2.3.7.5.6 The CPDLC-user-abort Service

2.3.7.5.6.1 Issuing a CPDLC-user-abort Request

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

2.3.7.6 Message Intent

2.3.7.6.1 Purpose

Note. — 2.3.7.6 contains the message set for CPDLC. Message attributes, message presentation guidance, and data structure presentation guidance are presented. The actual information exchanged between an aircraft and ground peer or a ground and ground peer CPDLC applications is defined in 2.3.4; however, 2.3.4 does not mandate any particular method for presenting this information. The presentation of information to the controller and aircraft crew is a local implementation. The message presentation recommendations contained in Tables 2.3.7-5 to 2.3.7-28 are one possible means of presenting the information. These recommendations are generally consistent with current ICAO practices for displaying ATC information.

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

Table 2.3.7-5. Responses/Acknowledgments (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
0	Indicates that ATS cannot comply with the request.	UNABLE	N	M	N
1	Indicates that ATS has received the message~~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 message~~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 2.3.7-6. Vertical Clearances (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Notification that an level~~altitude~~ change instruction should be expected.	EXPECT [level~~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 level~~altitude~~.	AT [time] EXPECT CLIMB TO [level~~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 level~~altitude~~.	AT [position] EXPECT CLIMB TO [level~~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 level~~altitude~~.	AT [time] EXPECT DESCENT TO [level~~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 level~~altitude~~.	AT [position] EXPECT DESCENT TO [level~~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 level~~altitude~~.	AT [time] EXPECT CRUISE CLIMB TO [level~~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 level~~altitude~~.	AT [position] EXPECT CRUISE CLIMB TO [level~~altitude~~]	L	L	R
19	Instruction to maintain the specified level~~altitude~~.	MAINTAIN [level~~altitude~~]	N	M	W/U
20	~~Combined i~~Instruction that a climb to a specified level~~altitude~~ is to commence and the level~~altitude~~ is to be maintained when reached.	CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~]	N	M	W/U

21	~~Combined i~~Instruction that at the specified time, a climb to the specified level~~altitude~~ is to commence and once reached the specified level~~altitude~~ is to be maintained.	AT [time] CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~]	N	M	W/U
22	~~Combined i~~Instruction that at the specified position, a climb to the specified level~~altitude~~ is to commence and once reached the specified level~~altitude~~ is to be maintained.	AT [position] CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~]	N	M	W/U
185	~~Combined i~~Instruction that after passing the specified position, a climb to the specified level~~altitude~~ is to commence and once reached the specified level~~altitude~~ is to be maintained.	AFTER PASSING [position] CLIMB TO ~~AND MAINTAIN~~ [level~~altitude~~]	N	M	W/U
23	~~Combined i~~Instruction that a descent to a specified level~~altitude~~ is to commence and the level~~altitude~~ is to be maintained when reached.	DESCEND TO AND MAINTAIN [level~~altitude~~]	N	M	W/U
24	~~Combined i~~Instruction that at a specified time a descent to a specified level~~altitude~~ is to commence and once reached the specified level~~altitude~~ is to be maintained.	AT [time] DESCEND TO AND MAINTAIN [level~~altitude~~]	N	M	W/U
25	~~Combined i~~Instruction that at the specified position a descent to the specified level~~altitude~~ is to commence and when the specified level~~altitude~~ is reached it is to be maintained.	AT [position] DESCEND TO AND MAINTAIN [level~~altitude~~]	N	M	W/U
186	~~Combined i~~Instruction that after passing the specified position, a descent to the specified level~~altitude~~ is to commence and once reached the specified level~~altitude~~ is to be maintained.	AFTER PASSING [position] DESCEND TO AND MAINTAIN [level~~altitude~~]	N	M	W/U
26	~~Combined i~~Instruction that a climb is to commence at a rate such that the specified level~~altitude~~ is reached at or before the specified time.	CLIMB TO REACH [level~~altitude~~] BY [time]	N	M	W/U
27	~~Combined i~~Instruction that a climb is to commence at a rate such that the specified level~~altitude~~ is reached at or before the specified position.	CLIMB TO REACH [level~~altitude~~] BY [position]	N	M	W/U
28	~~Combined i~~Instruction that a descent is to commence at a rate such that the specified level~~altitude~~ is reached at or before the specified time.	DESCEND TO REACH [level~~altitude~~] BY [time]	N	M	W/U
29	~~Combined i~~Instruction that a descent is to commence at a rate such that the specified level~~altitude~~ is reached at or before the specified position.	DESCEND TO REACH [level~~altitude~~] BY [position]	N	M	W/U
192	~~Combined i~~Instruction that a change of level~~altitude~~ is to continue, but at a rate such that the specified level~~altitude~~ is reached at or before the specified time.	REACH [level~~altitude~~] BY [time]	N	M	W/U

209	~~Combined i~~Instruction that a change of level~~altitude~~ is to continue, but at a rate such that the specified level~~altitude~~ is reached at or before the specified position	REACH [level~~altitude~~] BY [position]	N	M	W/U
30	~~An~~ level~~altitude~~ within the defined ~~block~~ vertical range~~altitude~~ specified is to be maintained.	MAINTAIN BLOCK [level~~altitude~~] TO [level~~altitude~~]	N	M	W/U
31	~~Combined i~~Instruction that a climb to an level~~altitude~~ within the vertical range ~~block altitude~~ defined is to commence.	CLIMB TO AND MAINTAIN BLOCK [level~~altitude~~] TO [level~~altitude~~]	N	M	W/U
32	~~Combined i~~Instruction that a descent to an level~~altitude~~ within the vertical~~the~~ range ~~block altitude~~ defined is to commence.	DESCEND TO AND MAINTAIN BLOCK [level~~altitude~~] TO [level~~altitude~~]	N	M	W/U
34	A cruise climb is to commence and continue until the specified level~~altitude~~ is reached.	CRUISE CLIMB TO [level~~altitude~~]	N	M	W/U
35	A cruise climb can commence once above the specified level~~altitude~~.	CRUISE CLIMB ABOVE [level~~altitude~~]	N	M	W/U
219	Instruction to stop the climb below the previously assigned level~~altitude~~.	STOP CLIMB AT [level~~altitude~~]	U	M	W/U
220	Instruction to stop the descent above the previously assigned level~~altitude~~.	STOP DESCENT AT [level~~altitude~~]	U	M	W/U
36	The climb to the specified level~~altitude~~ should be made at the aircraft's best rate.	EXPEDITE CLIMB TO [level~~altitude~~]	U	M	W/U
37	The descent to the specified level~~altitude~~ should be made at the aircraft's best rate.	EXPEDITE DESCENT TO [level~~altitude~~]	U	M	W/U
38	Urgent instruction to immediately climb to the specified level~~altitude~~.	IMMEDIATELY CLIMB TO [level~~altitude~~]	D	H	W/U
39	Urgent instruction to immediately descend to the specified level~~altitude~~.	IMMEDIATELY DESCEND TO [level~~altitude~~]	D	H	W/U
40	Urgent instruction to immediately stop a climb once the specified level~~altitude~~ is reached.	IMMEDIATELY STOP CLIMB AT [level~~altitude~~]	D	H	W/U
41	Urgent instruction to immediately stop a descent once the specified level~~altitude~~ is reached.	IMMEDIATELY STOP DESCENT AT [level~~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
33	*Reserved.*~~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 2.3.7-7. Crossing Constraints (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
42	Notification that an levelaltitude change instruction should be expected which will require the specified position to be crossed at the specified levelaltitude.	EXPECT TO CROSS [position] AT [levelaltitude]	L	L	R
43	Notification that an levelaltitude change instruction should be expected which will require the specified position to be crossed at or above the specified levelaltitude.	EXPECT TO CROSS [position] AT OR ABOVE [levelaltitude]	L	L	R
44	Notification that an levelaltitude change instruction should be expected which will require the specified position to be crossed at or below the specified levelaltitude.	EXPECT TO CROSS [position] AT OR BELOW [levelaltitude]	L	L	R
45	Notification that an levelaltitude change instruction should be expected which will require the specified position to be crossed at the specified levelaltitude which is to be maintained subsequently.	EXPECT TO CROSS [position] AT AND MAINTAIN [levelaltitude]	L	L	R
46	The specified position is to beshould be crossed at the specified levelaltitude. This may require the aircraft to modify its climb or descent profile.	CROSS [position] AT [levelaltitude]	N	M	W/U
47	The specified position is to be crossed at or above the specified levelaltitude.	CROSS [position] AT OR ABOVE [levelaltitude]	N	M	W/U
48	The specified position is to be crossed at or below the specified levelaltitude.	CROSS [position] AT OR BELOW [levelaltitude]	N	M	W/U
49	Combined iInstruction that the specified position is to be crossed at the specified levelaltitude and that levelaltitude is to be maintained when reached.	CROSS [position] AT AND MAINTAIN [levelaltitude]	N	M	W/U
50	The specified position is to be crossed at an levelaltitude between the specified levelaltitudes.	CROSS [position] BETWEEN [levelaltitude] AND [levelaltitude]	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 levelaltitude.	CROSS [position] AT [time] AT [levelaltitude]	N	M	W/U
59	The specified position is to be crossed at or before the specified time and at the specified levelaltitude.	CROSS [position] AT OR BEFORE [time] AT [levelaltitude]	N	M	W/U
60	The specified position is to be crossed at or after the specified time and at the specified levelaltitude.	CROSS [position] AT OR AFTER [time] AT [levelaltitude]	N	M	W/U
61	Combined iInstruction that the specified position is to be crossed at the specified levelaltitude and speed and the levelaltitude and speed are to be maintained.	CROSS [position] AT AND MAINTAIN [levelaltitude] AT [speed]	N	M	W/U
62	Combined iInstruction that at the specified time the specified position is to be crossed at the specified levelaltitude and the levelaltitude is to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [levelaltitude]	N	M	W/U
63	Combined iInstruction that at the specified time the specified position is to be crossed at the specified levelaltitude and speed and the levelaltitude and speed are to be maintained.	AT [time] CROSS [position] AT AND MAINTAIN [levelaltitude] AT [speed]	N	M	W/U

Table 2.3.7-8. Lateral Offsets (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
64	Instruction to fly a parallel track to the cleared route at a displacement of the specified distance in the specified direction.	OFFSET [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 2.3.7-9. Route Modifications (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
73	Notification to the aircraft of the instructions to be followed from departure until the specified clearance limit.	[departure clearance]	N	M	W/U
74	Instruction to proceed directly from its present position to the specified position.	PROCEED DIRECT TO [position]	N	M	W/U
75	Instruction to proceed, when able, directly to the specified position.	WHEN ABLE PROCEED DIRECT TO [position]	N	M	W/U
76	Instruction to proceed, at the specified time, directly to the specified position.	AT [time] PROCEED DIRECT TO [position]	N	M	W/U
77	Instruction to proceed, at the specified position, directly to the next specified position.	AT [position] PROCEED DIRECT TO [position]	N	M	W/U
78	Instruction to proceed, upon reaching the specified level~~altitude~~, directly to the specified position.	AT [level~~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
236	Instruction to leave controlled~~proceed to the intended destination which has no instrument approach procedure authorized or available, and is in uncontrolled~~ airspace.	LEAVE~~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 level~~altitude~~ is reached may be issued.	AT [level~~altitude~~] EXPECT DIRECT TO [position]	L	L	R
91	Instruction to enter a holding pattern with the specified characteristics at the specified position and level~~altitude~~.	HOLD AT [position] MAINTAIN [level~~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 level~~altitude~~.	HOLD AT [position] AS PUBLISHED MAINTAIN [level~~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 2.3.7-10. Speed Changes (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
100	Notification that a speed instruction may be issued to be effective at the specified time.	AT [time] EXPECT [speed]	L	L	R
101	Notification that a speed instruction may be issued to be effective at the specified position.	AT [position] EXPECT [speed]	L	L	R
102	Notification that a speed instruction may be issued to be effective at the specified level~~altitude~~.	AT [level~~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 level~~altitude~~.	AT [level~~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	W/U

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

	Message Intent/Use	Message Element	URG	ALRT	RESP
117	The ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	CONTACT [~~icao~~unitname] [frequency]	N	M	W/U
118	At the specified position the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [position] CONTACT [~~icao~~unitname] [frequency]	N	M	W/U
119	At the specified time the ATS unit with the specified ATS unit name is to be contacted on the specified frequency.	AT [time] CONTACT [~~icao~~unitname] [frequency]	N	M	W/U
120	The ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	MONITOR [~~icao~~unitname] [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 [~~icao~~unitname] [frequency]	N	M	W/U
122	At the specified time the ATS unit with the specified ATS unit name is to be monitored on the specified frequency.	AT [time] MONITOR [~~icao~~unitname] [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 level~~altitude~~ information.	SQUAWK MODE CHARLIE	N	M	W/U
126	The SSR transponder responses should no longer include level~~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 2.3.7-12. Report/Confirmation Requests (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
127	Instruction to report when the aircraft is back on the cleared route.	REPORT BACK ON ROUTE	N	L	W/U~~R~~
128	Instruction to report when the aircraft has left the specified level~~altitude~~.	REPORT LEAVING [level~~altitude~~]	N	L	W/U~~R~~
129	Instruction to report when the aircraft is in level~~altitude~~ flight at the specified level~~altitude~~.	REPORT MAINTAINING~~WHEN LEVEL AT~~ [level~~altitude~~]	N	L	W/U~~R~~
175	Instruction to report when the aircraft has reached the specified level~~altitude~~.	REPORT REACHING [level~~altitude~~]	N	L	W/U~~R~~
180	Instruction to report when the aircraft is within the specified vertical~~altitude~~ range.	REPORT REACHING BLOCK [level~~altitude~~] TO [level~~altitude~~]	N	L	W/U~~R~~
130	Instruction to report when the aircraft has passed the specified position.	REPORT PASSING [position]	N	L	W/U~~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 ON BOARD	U	M	Y
132	Instruction to report the present position.	REPORT POSITION	N	M	Y
133	Instruction to report the present level~~altitude~~.	REPORT PRESENT LEVEL~~ALTITUDE~~	N	M	Y
134	Instruction to report the requested speed.	REPORT [speed type] [speed type] [speed type] SPEED	N	M	Y
135	Instruction to confirm and acknowledge the currently assigned level~~altitude~~.	CONFIRM ASSIGNED LEVEL~~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 (SSR).	CONFIRM SQUAWK	N	L	Y
145	Instruction to report the present heading.	REPORT HEADING	N	M	Y
146	Instruction to report the present ground track.	REPORT GROUND TRACK	N	M	Y
182	Instruction to report the identification code of the last ATIS received.	CONFIRM ATIS CODE	N	L	Y
147	Instruction to make a position report.	REQUEST POSITION REPORT	N	M	Y
216	Instruction to file a flight plan.	REQUEST FLIGHT PLAN	N	M	Y
217	Instruction to report that the aircraft has landed.	REPORT ARRIVAL	N	M	Y
229	Instruction to report the preferred alternate aerodrome for landing.	REPORT ALTERNATE AERODROME	L	L	Y
231	Instruction to indicate the pilot's preferred level~~altitude~~.	STATE PREFERRED LEVEL~~ALTITUDE~~	L	L	Y
232	Instruction to indicate the pilot's preferred time and/or position to commence descent to the aerodrome of intended arrival.	STATE TOP OF DESCENT	L	L	Y

Table 2.3.7-13. Negotiation Requests (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
148	Request for the earliest time at which the specified level~~altitude~~ can be accepted.	WHEN CAN YOU ACCEPT [level~~altitude~~]	N	L	Y
149	Instruction to report whether or not the specified level~~altitude~~ can be accepted at the specified position.	CAN YOU ACCEPT [level~~altitude~~] AT [position]	N	L	A/N
150	Instruction to report whether or not the specified level~~altitude~~ can be accepted at the specified time.	CAN YOU ACCEPT [level~~altitude~~] AT [time]	N	L	A/N
151	Instruction to report the earliest time when the specified speed~~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 2.3.7-14. Air Traffic Advisories (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
153	ATS advisory that the altimeter setting should be the specified setting.	ALTIMETER [altimeter]	N	L	R
213	ATS advisory that the specified altimeter setting relates to the specified facility.	[~~icao~~ facility designation~~or~~] 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 designation~~or~~] 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 2.3.7-15. System Management Messages (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 designation~~or~~]	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 designation~~or~~]	L	N	N
227	Confirmation to the aircraft system that the ground system has received the message to which the logical acknowledgment refers and found it acceptable for display to the responsible person.	LOGICAL ACKNOWLEDGMENT	N	M	N
233	Notification to the pilot that messages sent requiring a logical acknowledgment will not be accepted by this ground system.	USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED	N	M	N

Table 2.3.7-16. Additional Messages (uplink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
164	The associated instruction may be complied with at any future time.	WHEN READY	L	N	N
230	The associated instruction is to be complied with immediately.	IMMEDIATELY	D	H	N
165	Used to link two messages, indicating the proper order of execution of clearances/ instructions.	THEN	L	N	N
166	The associated instruction is issued due to traffic considerations.	DUE TO [traffic type] TRAFFIC	L	N	N
167	The associated instruction is issued due to airspace restrictions.	DUE TO AIRSPACE RESTRICTION	L	N	N
168	The indicated communication should be ignored.	DISREGARD	U	M	R
176	Notification that the operator is responsible for maintaining separation from other traffic and is also responsible for maintaining Visual Meteorological Conditions.	MAINTAIN OWN SEPARATION AND VMC	N	M	W/U
177	Used in conjunction with a clearance/instruction to indicate that the operator may execute when prepared to do so.	AT PILOTS DISCRETION	L	L	N
169		[free text]	N	L	R
170		[free text]	D	H	R
194		[free text]	N	L	Y
178		[free text]	N	L	N
195		[free text]	L	L	R
196		[free text]	N	M	W/U
197		[free text]	U	M	W/U
198		[free text]	D	H	W/U
199		[free text]	N	M	W/U
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

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

Table 2.3.7-17. Responses (downlink)

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

Table 2.3.7-18. Vertical Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
6	Request to fly at the specified levelaltitude.	REQUEST [levelaltitude]	N	L	Y
7	Request to fly at an levelaltitude within the specified vertical rangealtitude interval.	REQUEST BLOCK [levelaltitude] TO [levelaltitude]	N	L	Y
8	Request to cruise climb to the specified levelaltitude.	REQUEST CRUISE CLIMB TO [levelaltitude]	N	L	Y
9	Request to climb to the specified levelaltitude.	REQUEST CLIMB TO [levelaltitude]	N	L	Y
10	Request to descend to the specified levelaltitude.	REQUEST DESCENT TO [levelaltitude]	N	L	Y
11	Request that at the specified position a climb to the specified levelaltitude be approved.	AT [position] REQUEST CLIMB TO [levelaltitude]	N	L	Y
12	Request that at the specified position a descent to the specified levelaltitude be approved.	AT [position] REQUEST DESCENT TO [levelaltitude]	N	L	Y
13	Request that at the specified time a climb to the specified levelaltitude be approved.	AT [time] REQUEST CLIMB TO [levelaltitude]	N	L	Y
14	Request that at the specified time a descent to the specified levelaltitude be approved.	AT [time] REQUEST DESCENT TO [levelaltitude]	N	L	Y
69	Request that a descent be approved on a see-and-avoid basis.	REQUEST VMC DESCENT	N	L	Y

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

	Message Intent/Use	Message Element	URG	ALRT	RESP
15	Request that a parallel track, offset from the cleared track by the specified distance in the specified direction, be approved.	REQUEST OFFSET [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 2.3.7-20. Speed Requests (downlink)

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

Table 2.3.7-21. Voice Contact Requests (downlink)

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

Table 2.3.7-22. Route Modification Requests (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
22	Request to track from the present position direct to the specified position.	REQUEST DIRECT TO [position]	N	L	Y
23	Request for the specified procedure clearance.	REQUEST [procedure name]	N	L	Y
24	Request for a route clearance.	REQUEST CLEARANCE [route clearance]	N	L	Y
25	Request for a clearance.	REQUEST [clearance type] CLEARANCE	N	L	Y
26	Request for a weather deviation to the specified position via the specified route.	REQUEST WEATHER DEVIATION TO [position] VIA [route clearance]	N	M	Y
27	Request for a weather deviation up to the specified distance off track in the specified direction.	REQUEST WEATHER DEVIATION UP TO [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 2.3.7-23. Reports (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
28	Notification of leaving the specified levelaltitude.	LEAVING [levelaltitude]	N	L	N
29	Notification of climbing to the specified levelaltitude.	CLIMBING TO [levelaltitude]	N	L	N
30	Notification of descending to the specified levelaltitude.	DESCENDING TO [levelaltitude]	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 levelaltitude.	PRESENT LEVELALTITUDE[levelaltitude]	N	L	N
33	Notification of the present position.	PRESENT POSITION [position]	N	L	N
34	Notification of the present speed.	PRESENT SPEED [speed]	N	L	N
113	Notification of the requested speed type.	[speed type] [speed type] [speed type] SPEED [speed]	N	L	N
35	Notification of the present heading in degrees.	PRESENT HEADING [degrees]	N	L	N
36	Notification of the present ground track in degrees.	PRESENT GROUND TRACK [degrees]	N	L	N
37	Notification that the aircraft is maintaining the specified levelaltitude.	LEVELALTITUDE [levelaltitude]	N	L	N
72	Notification that the aircraft has reached the specified levelaltitude.	REACHING [levelaltitude]	N	L	N
76	Notification that the aircraft has reached an levelaltitude within the specified verticalaltitude range.	REACHING BLOCK [levelaltitude] TO [levelaltitude]	N	L	N
38	Read-back of the assigned levelaltitude.	ASSIGNED LEVELALTITUDE[levelaltitude]	N	M	N
77	Read-back of the assigned verticalaltitude range.	ASSIGNED BLOCK [levelaltitude] TO [levelaltitude]	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 [~~icao~~unitname] [frequency]	U	M	N
102	Used to report that an aircraft has landed.	LANDING REPORT	N	N	N
104	Notification of estimated time of arrival at the specified position.	ETA [position][time]	L	L	N
105	Notification of the alternative aerodrome for landing.	ALTERNATE AERODROME [airport]	L	L	N
106	Notification of the preferred levelaltitude	PREFERRED LEVEL~~ALTITUDE~~ [levelaltitude]	L	L	N
109	Notification of the preferred time to commence descent for approach	TOP OF DESCENT [time]	L	L	N
110	Notification of the preferred position to commence descent for approach	TOP OF DESCENT [position]	L	L	N
111	Notification of the preferred time and position to commence descent for approach	TOP OF DESCENT [time] position]	L	L	N

Table 2.3.7-24. Negotiation Requests (downlink)

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

Table 2.3.7-25. Emergency Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
55	Urgency prefix.	PAN PAN PAN	U	H	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 on board] PERSONS 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.	OFFSETTING [distance offset] [direction] OF ROUTE	U	H	N
61	Notification that the aircraft is descending to the specified level~~altitude~~.	DESCENDING TO [level~~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 2.3.7-26. System Management Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
62	A system generated message that the avionics has detected an error.	ERROR [error information]	U	L	N
63	A system generated denial to any CPDLC message sent from a ground facility that is not the Current Data Authority.	NOT CURRENT DATA AUTHORITY	L	L	N
99	A system generated message to inform a ground facility that it is now the Current Data Authority	CURRENT DATA AUTHORITY	L	L	N
107	A system generated message sent to a ground system that tries to connect to an aircraft when a current data authority has not designated the ground system as the NDA.	NOT AUTHORIZED NEXT DATA AUTHORITY	L	L	N
64	Notification to the ground system that the specified ATSU is the current data authority.	[~~icao~~ facility designation~~or~~]	L	L	N
73	A system generated message indicating the software version number.	[version number]	L	L	N
100	Notification to the ground system that the aircraft system has received the message to which the logical acknowledgment refers.	LOGICAL ACKNOWLEDGMENT	N	M	N

Table 2.3.7-27. Additional Messages (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
65	Used to explain reasons for aircraft operator's message.	DUE TO WEATHER	L	L	N
66	Used to explain reasons for aircraft operator's message.	DUE TO AIRCRAFT PERFORMANCE	L	L	N
74	States a desire by the aircraft operator to provide his/her own separation and remain in VMC~~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 2.3.7-28. Negotiation Responses (downlink)

	Message Intent/Use	Message Element	URG	ALRT	RESP
81	We can accept the specified level~~altitude~~ at the specified time.	WE CAN ACCEPT [level~~altitude~~] AT [time]	L	L	N
82	We cannot accept the specified level~~altitude~~.	WE CANNOT ACCEPT [level~~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] [direction] 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] [direction]	L	L	N

2.3.7.7 Parameter Value Unit, Range, and Resolution

2.3.7.7.1 A CPDLC-user shall interpret CPDLC message element variables unit, range, and resolution as defined in 2.3.4.

2.3.8. SUBSETTING RULES

2.3.8.1 General

Note. — This chapter specifies conformance requirements which all implementations of the CPDLC protocol obey.

2.3.8.1.1 An implementation of either the CPDLC ground based service or the CPDLC air based service claiming conformance to 2.3 shall support the CPDLC protocol features as shown in the tables below:

Note. — The 'status' column indicates the level of support required for conformance to the CPDLC-ASE protocol described in this part. The values are as follows:

'M' *mandatory support is required,*

'O' *optional support is permitted for conformance to the CPDLC protocol,*

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

'C.n' *the item is conditional where n is the number which identifies the condition which is applicable. The definitions for the conditional statements used in this chapter are written under the tables in which they first appear.*

Table 2.3.2.3.8-1. Protocol Versions Implemented

	Status	Associated Predicate
Version 1	M	none

Table 2.3. 2.3.8-2. CPDLC Protocol Options

	Status	Associated Predicate
CPDLC-air-ASE	C.1	CPDLC/air
CPDLC-ground-ASE	C.1	CPDLC/ground
DSC function supported	if (CPDLC/air) O, else M	DSC-FU
DSC function supported by CPDLC-ground-user	if (CPDLC/ground) O, else N/A	DSC-USER
Forward function supported by initiating user	if (CPDLC/ground) O, else N/A	FWD-INIT
Forward function supported by receiving user	if (CPDLC/ground) O, else N/A	FWD-USER

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

Table 2.3.8-3. CPDLC-air-ASE Conformant Configurations

	List of Predicates	Functionality Description
I.	CPDLC/air	a CPDLC-air-ASE supporting just the core* CPDLC functionality (no downstream clearance capability)
II.	CPDLC/air + DSC-FU	a CPDLC-air-ASE supporting the core CPDLC functionality and the downstream clearance capability (complete CPDLC-air-ASE functionality)
		* the core CPDLC functionality is defined as support for the CPDLC-start, message, end services plus abort services.

Table 2.3.8-4. CPDLC-ground-ASE Conformant Configurations

	List of Predicates	Functionality Description
I.	CPDLC/ground ~~+ DSC-FU~~	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for receiving and indicating that the forward function is not supported
II.	CPDLC/ground + DSC-FU + DSC-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for receiving and indicating that the ground forward function is not supported
III.	CPDLC/ground + DSC-FU + FWD-INIT	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for receiving and indicating that the ground forward function is not supported • functionality for supporting the capability to initiate the ground forwarding of CPDLC messages
IV.	CPDLC/ground + DSC-FU + DSC-USER + FWD-INIT	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for receiving and indicating that the ground forward function is not supported • functionality for supporting the capability to initiate the ground forwarding of CPDLC messages
V.	CPDLC/ground + DSC-FU + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • functionality for CPDLC-ground user support for the receipt of CPDLC ground forward messages
VI.	CPDLC/ground + DSC-FU + DSC-USER + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • functionality for CPDLC-ground user support for the receipt of CPDLC ground forward messages
VII.	CPDLC/ground + DSC-FU + FWD-INIT + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality plus: • functionality for receiving and CPDLC-ground-user rejecting a request for a DSC dialogue • full CPDLC ground forwarding functionality (initiating and user receiving)
VIII.	CPDLC/ground + DSC-FU + DSC-USER + FWD-INIT + FWD-USER	a CPDLC-ground-ASE supporting the core CPDLC functionality and downstream clearance function plus • full CPDLC ground forwarding functionality (initiating and user receiving) (complete CPDLC-ground-ASE functionality)

Table 2.3. 8-5. Supported CPDLC Service Primitives

	Sending (req,[cnf])	Receiving (ind, [rsp])
CPDLC-start service	M	M
CPDLC-message service	M	M
CPDLC-end service	if (CPDLC/ground) M, else N/A	if (CPDLC/air) M, else N/A
DSC-start service	if (CPDLC/air and DSC-FU) M, else N/A	if (CPDLC/ground) M, else N/A
DSC-end service	if (CPDLC/air and DSC-FU) M, else N/A	if (CPDLC/ground and DSC-USER) M, else N/A
CPDLC-forward service	if (FWD-INIT) M, else N/A	if (FWD-USER) M, else N/A
CPDLC-user-abort	M	M
CPDLC-provider-abort	N/A	M

2.3.8.2 Table 2.3. 8-6. Supported CPDLC APDUs

	Sender	**Receiver**
[GroundPDUs] startup	if (CPDLC/ground) M, else N/A	if (CPDLC/air) M, else N/A
[GroundPDUs] send	if (CPDLC/ground) M, else N/A	if (CPDLC/air) M, else N/A
[GroundPDUs] forward	if (FWD-INIT) M, else N/A	if (CPDLC/ground) M, else N/A
[GroundPDUs] forwardresponse	if (CPDLC/ground) M, else N/A	if (FWD-INIT) M, else N/A
[GroundPDUs] abort	if (CPDLC/ground) M, else N/A	if (CPDLC/air or FWD-INIT) M, else N/A
[AircraftPDUs] startdown	if (CPDLC/air) M, else N/A	if (CPDLC/ground) M, else N/A
[AircraftPDUs] send	if (CPDLC/air) M, else N/A	if (CPDLC/ground) M, else N/A
[AircraftPDUs] abort	if (CPDLC/air) M, else N/A	if (CPDLC/ground) M, else N/A