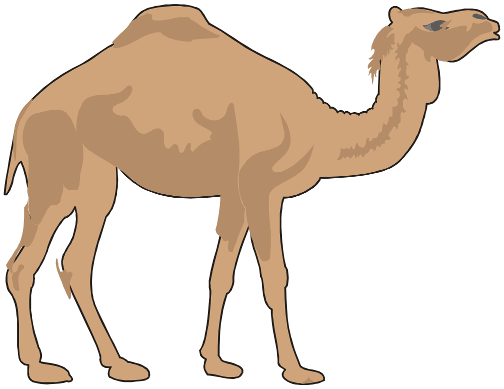
Aeronautical Telecommunication Network (ATN)

Comprehensive ATN Manual (CAMAL)

Part III
Applications Guidance Material



Editor's Draft (January 1999)

The preparation of this document has been on a "best efforts" basis and no warrantee is offered as to its correctness..

This PDF version has been prepared for the ATNP Working Groups by FANS Information Services Ltd - http://www.fans-is.com
Please check our Web Site regularly for information on updates to the SARPs and Guidance Material

9th January 1999

PART III

TABLE OF CONTENTS

1. *INTRODUCTION*

1.1 This part of the document contains guidance material on all air-ground and ground-ground applications covered by the first set of Standards and Recommended Practices (SARPs) for the aeronautical telecommunication network (ATN) (commonly known as the CNS/ATM-1 Package).

2. CM APPLICATION

2.1 Introduction

2.1.1 Purpose

2.1.1.1 In line with normal ICAO practice, this chapter of Part III was developed as a companion document to the Context Management (CM) Standards and Recommended Practices (SARPs) ([1]). It may be read alongside the CM SARPs, in order to provide a greater understanding of the specification itself, or it may be read instead of the CM SARPs by readers that simply want to understand the purpose of the CM Application rather than the detail of the specification.

2.1.1.2 This chapter also provides some historical information on the development of CM and explanations as to why CM is specified in the SARPs the way that it is, including further explanation of notes and recommendations.

2.1.2 Scope

2.1.2.1 This chapter of provides guidance material for those implementing the CM Application.

2.1.2.2 This chapter of does not define any mandatory or optional requirements for CM, neither does it define any recommended practices. This chapter of is not intended to instruct users on how to use CM in a particular operational environment.

2.1.2.3 The CM application SARPs are dedicated to Air Traffic Services (ATS). Aeronautical Operational Control (AOC) may choose to use the CM SARPs as a model for their own applications.

2.1.3 History

2.1.3.1 The CM application allows appropriately equipped aircraft and ground systems or two ground systems to exchange and update data link application information between each other. The Automatic Dependent Surveillance Panel (ADSP) has created new operational requirements necessary to fulfill the initial contact and application information exchange.

2.1.3.2 The functionality specified by the ADSP in the ICAO *Manual of Air Traffic Services Data Link Applications* (Doc 9694) ([3]) states that the Data Link Initiation Capability (DLIC) be aircraft initiated, and have the following capabilities:

a) Logon: data link application initiation and, if required, flight plan association,

b) Update: updating of previously coordinated initiation information,

c) Contact: instructions from a ground system to perform data link iniation with another specified ground system,

d) Registration: local dissemination of application information, and

e) Ground Forwarding: ground-ground forwarding of aircraft application information.

2.1.3.3 The CM SARPs have been developed in order to fulfill the DLIC requirements as set forth by [3]. ICAO has specified that the CM application conform to the ATN protocols for data link operations.

2.1.3.4 The initial CM concept was a system-level application information (i.e. application addresses, names and version numbers) exchange function between the aircraft and the ground. After an initial log-on message was sent by the aircraft requesting support for specific services, the ground system would reply, passing the necessary address information. Within the same uplinked message, the ground would terminate the dialogue. This left both the ground system and aircraft a list of application information which could be used as necessary. Once the initial application information was received by the ground system, the ground system also had capability to provide updated application information to an aircraft as well as to direct an aircraft to perform a logon to another ground system.

2.1.3.5 In the course of the development of these SARPs, the concept has become more complex as more operational capability was included. In particular, the ADSP identified a need for the ground system to maintain the dialogue after the initial connection had been made. The requirement for this facility applied particularly, but not exclusively, to aircraft operating in regions where Flight Information Regions (FIRs) can be small, and consequently the exchange of application information with relevant data authorities can be frequent.

2.1.3.6 There was also an explicit requirement in the ADSP material for the Air Traffic Service Unit (ATSU) which had received information from the CM logon service to be able to pass aircraft address information to an adjacent ATSU. The Aeronautical Telecommunication Network Panel (ATNP) therefore decided to include an element of ground/ground message forwarding in the CM functionality.

2.1.3.7 States may not be willing to incur the costs of implementing a complete system if they only ever intended to use certain elements of the application. For example, both the maintenance of a dialogue and the ground/ground functionality were seen by some States as having a limited applicability. The SARPs therefore took account of the need to separate out the functionalities to enable partial implementation, while still retaining the interoperability required by the ICAO Standards. This led to the development of subsetting rules, and the identification of conformant configurations.

2.1.3.8 The ATNP worked very closely with the ADSP to ensure that the development of both the operational concepts and the technical means of achieving them kept in step with each other. However, the ADSP generally looked at a longer timescale than the current ATNP initial implementation programme, and this will inevitably mean that some elements of their work has not been incorporated into the present SARPs.

2.1.3.9 The ATNP identified a set of packages to accommodate the continuous specification of operational requirements by the ADSP. This chapter of Part III provides guidance material for Version 1 of the CM application, contained in the first set of ATN SARPs (known as CNS/ATM-1 Package).

2.1.4 Structure

2.1.4.1 Chapter 2.1 — INTRODUCTION — This chapter contains the reason for providing guidance material as well as the scope. In addition, it provides a brief overview of the CM functionality, CM's relationship with other SARPs, and identifies applicable reference documents.

2.1.4.2 Chapter 2.2 — OVERALL GENERAL FUNCTIONALITY — This chapter describes generic concepts that are used throughout the CM SARPs and guidance material. This chapter also covers some implementation issues that are not addressed in the SARPs.

2.1.4.3 Chapter 2.3 — CM FUNCTIONAL DESCRIPTION — This chapter gives a functional breakdown of the various services that CM provides. It describes a peer to peer interaction, including reasons for why particular information is used or not used, and what actions on the information are expected.

2.1.4.4 Chapter 2.4 — CM SARPs SECTION DESCRIPTION — This chapter clarifies any functionality that was not addressed in Chapter 2.3 on a chapter by chapter basis.

2.1.4.5 Chapter 2.5 — DIMENSIONS — This chapter gives some sample encoding sizes for guidance on what capacities need to be allowed for in order to meet normal operational expectations.

2.1.4.6 Chapter 2.6 — INDEXES/TABLES — This chapter describes each abstract syntax notation one (ASN.1) field composing the CM protocol data units as well as provides a cross-reference between fields.

2.1.4.7 Chapter 2.7 — EXAMPLE SCENARIOS — This chapter gives some examples as to what typical scenarios one can expect in course of normal CM operation.

2.1.4.8 Chapter 2.8 — EXAMPLE ENCODING — This chapter outlines some actual sample packed coding rules (PER) encoding of typical CM messages.

2.1.5 CM Overview

2.1.5.1 This chapter gives a very brief, high level description of the CM as an application allowing an addressing capability for data link applications and contains an outline description of the functions which the CM application provides, namely:

a) The Logon Function — allows the exchange of application information,

b) The Update Function — allows a ground system to modify application data held by an aircraft,

c) The Contact Function — allows a ground system to direct an aircraft to logon to another ground system,

d) The Forward Function — allows a ground system to forward aircraft application information to another ground system, and

e) The Registration Function — allows an aircraft and ground system to make application information available to other applications or communications systems in the aircraft or ground system.

2.1.5.2 *Logon Function*

2.1.5.2.1 The logon function defines a method for the aircraft system to provide its data link application information along with additional flight plan information as required to a ground system. In response, the ground system provides its data link application information to the aircraft. The logon function is not restricted to logons solely with ground systems that provide ATC services; it may also be used to exchange and store addressing information to a centralized location. This is the central CM server concept which is further described in 2.2.2 of this chapter.

2.1.5.2.2 The logon function is used when the aircraft or ground system does not have a priori knowledge of the application information needed in order to initiate data link operations. If a ground system has the aircraft's CM address, it may use the update function in lieu of using the logon function (as detailed in 2.1.5.3 of this chapter).

2.1.5.2.3 The logon function refers to an application level (i.e. user) logon; the communications logon service is provided by the underlying communications protocols. Note that the logon function is not the equivalent of a logon to a computer system. A logon to a computer system provides the initial mechanism that allows all other functions to be performed. The logon as used in CM is strictly for the exchange, correlation and maintenance of data link application information, and is independent of other data link functions such as automatic dependent surveillance/flight information service (ADS/FIS) contract establishment or controller-pilot data link communications (CPDLC) dialogue initiation.

2.1.5.2.4 The logon function can be initiated by either the pilot or the avionics system. In order to invoke the logon function, some information must be provided by the avionics or pilot prior to invocation. This information includes:

a) The Initial Ground CM address, which is the address of the ground system’s CM application that is to be logged on to (some methods to obtain this address are detailed in 2.2.5 of this chapter). Note that this initial CM address is not actually part of the Logon Message, but is pre-requisite information that must be provided,

b) The Aircraft Flight ID, which is necessary in order for the receiving ground system to properly correlate the logon information to the correct flight plan,

c) The aircraft's CM Address, which is needed for update and ground forwarding purposes so that other end systems can communicate with the aircraft's CM application,

d) The application name and version numbers for all air-only initiated services that can be supported. This information is required so the ground system can determine version number compatibility for the applications the aircraft may wish to invoke. Addresses do not need to be provided by the aircraft, since the ground system cannot initiate the request of an air-only initiated service. Up to 256 applications can be specified. Note that if an aircraft does not choose to support a particular service, it need not provide the application name and version number of that service,

e) The application name, version number and address for all ground-initiated services that can be supported. This information is required so the ground system can both determine version number compatiblity and know the addresses for the specified aircraft applications. Note that applications that can be both ground and air initiated need to have their addresses provided for the case of ground initiation. Also, if an aircraft does not choose to support a particular service, it need not provide the application name, version number and address of that service. However, this may preclude an aircraft from operating in a particular environment. Up to 256 applications can be specified,

f) Other flight information as required, which is provided to allow unambiguous association of the aircraft with flight plan information stored on the ground. The other flight information as required refers to the facility designation, departure and destination airports, and date/time of departure. All of this information is optional. The facility designation may be provided to identify to a CM server which facility the aircraft wishes to perform data link functions with (this is discussed further in 2.2.2 of this chapter, Topology). The departure and destination airports and the date/time of departure may be provided to distinguish the aircraft where repetitive flight plans are used. Note that the date component of the data/time departure is not currently used operationally. If an operationally meaningful date is used and system or user action is intended upon receipt of that date, there has to be an agreement for such between both the aircraft and ground systems and users. Since this may not be the case in all operational environments, it is recommended that the system date or a pre-set, operationally meaningless date value which conforms to the date format be used,

g) The Called Facility Designation and the calling 24-bit Aircraft Address. These must be supplied for use by the dialogue service provider,

h) Class of Communication Service, which is a user option that specifies if there is a required network performance.

2.1.5.2.5 Upon receipt of the logon request, the ground system checks for CM application version number compatibility. If the version numbers are incompatible, the ground system will not accept the logon. If the version numbers are compatible, the ground system makes the information contained in the logon request available to the other applications within the ground system.

2.1.5.2.6 If the version number of the aircraft's CM application is compatible but different than the version number of the ground system's CM application, then consideration of the different versions must be taken into account when using CM features that are not backwards compatible.

2.1.5.2.7 Using the information contained in the logon request, the ground system establishes a correlation between the aircraft and the appropriate flight plan held on the ground. This ensures that there is an unambiguous mapping of the aircraft to its flight plan and that the ground system is communicating with the intended aircraft. If the information in the logon request is not sufficient for correlation, other means to establish the correlation should be used.

2.1.5.2.8 The addressing information contained in the logon request needs to be made available to the other applications (such as ADS, CPDLC, FIS, etc). The dialogue service provider will also need the application information for use in naming and addressing.

2.1.5.2.9 The ground system then responds to the logon request with a logon response. The ground user is recommeded to respond to the logon request with a logon response within 0.5 seconds. The logon response contains:

a) The application names, addresses and version numbers for the requested applications that can be air-initiated for all versions that the ground and aircraft can support. If the ground system does not choose to support a particular service, it need not provide the application name, version number and address of that service. Also, if an aircraft does not support a particular application (i.e. the application information is not contained in a logon request), then the corresponding ground system information for that application should not be returned to the aircraft. Up to 256 applications can be specified,

b) The application names and version numbers of the ground-initiated applications that are supported. The addresses do not need to be relayed to the aircraft since the ground is the only allowed initiator. If the ground system does not choose to support a particular service, it need not provide the application name and version number of that service. Also, if an aircraft does not support a particular application (i.e. the application information is not contained in a logon request), then the corresponding ground system information for that application should not be returned to the aircraft. Up to 256 applications can be specified, and

c) An indication on whether or not a dialogue is to be maintained.

2.1.5.2.10 Upon receipt of a logon response, the aircraft must first determine whether or not the logon was successful by checking to see if the version numbers were compatible. If the logon failed because the version numbers were not compatible, but a compatible version of CM exists on board the aircraft, then another logon using that version number should be attempted. If a compatible version of the CM application is not available, then local procedures must be executed in order to determine the level of service that can be given to the aircraft in its present environment.

2.1.5.2.11 If the logon was successful, the avionics must make the information contained in the logon response available to the other applications (such as ADS, CPDLC, FIS, etc) supported by the aircraft. The dialogue service provider will also need the application information for use in naming and addressing.

2.1.5.3 *Update Function*

2.1.5.3.1 This function provides the capability for the ground system to update application information contained in an aircraft system. This can be used if addressing or application information on the ground has changed (e.g. CPDLC is now unavailable) or in lieu of a logon if the aircraft's application and addressing information has been received by another means (e.g. ground-ground forwarding).

2.1.5.3.2 Generally the update function is invoked after a logon has been performed, so the ground system is sure to have the correct aircraft address. However, if a ground system has the correct aircraft information and a logon has not been performed, there is nothing to preclude the ground system from using the update function to provide the aircraft with its ground application information.

2.1.5.3.3 If an aircraft receives updated application information from the same source where data link services are already in place (i.e. an aircraft performing FIS with a ground system receives updated FIS application information from that ground system), the application information will be replaced. This may cause the data link service to terminate, and will have to subsequently be re-started.

2.1.5.3.4 The ground system must provide the following information for the update function:

a) The application names, addresses and version numbers for applications that can be air-initiated. If the ground system does not choose to support a particular service, it need not provide the application name, version number and address of that service. Up to 256 applications can be specified,

b) The application names and version numbers of the ground-initiated applications that are supported. If the ground system does not choose to support a particular service, it need not provide the application name and version number of that service. Up to 256 applications can be specified,

c) The called aircraft address and calling facility designation (if a dialogue does not exist), and

d) An indication of the Class of Communication required (if a dialogue does not exist).

2.1.5.3.4.1 If the aircraft is maintaining information from the sending ground facility of the update information, then the aircraft will update only that information of the sending ground facility (i.e., other application information that is held for other ground systems are not affected).

2.1.5.3.5 If the aircraft does not have information from the sending ground facility of the update information, then the aircraft will treat information the same as it would for a logon response (i.e. make the information contained in the update available to other applications and the dialogue service provider).

2.1.5.4 ***Contact Function***

2.1.5.4.1 This provides the capability for the ground system to request the aircraft to initiate the logon function with another designated ground system. If ground-ground connectivity does not exist between a ground system that has an aircraft's application information and a ground system that does not, the contact function can provide a means to provide the aircraft's data link application information to the ground system in need. If ground-ground connectivity exists between a ground system that has an aircraft's application information and a ground system that does not, the ground-ground forward function may be used to pass the aircraft's application information to the ground system in need, and the contact function does not have to be used (see 2.1.5.5 of this chapter).

2.1.5.4.2 The ground system must provide the following information for the contact function:

a) The called aircraft address and calling facility designation (if a dialogue doesn't already exist) , to be used by the dialogue service provider for addressing purposes,

b) An indication of the Class of Communication (if a dialogue doesn't already exist),

c) The facility designation of the ground system that the aircraft is to contact, and

d) The CM address of the ground system that the aircraft is to contact.

2.1.5.4.3 Upon receipt of a contact request, the aircraft will perform a logon function (as detailed in 2.1.5.2 of this chapter) with the ground system identified in the contact request. The aircraft is recommended to initiate the logon request with the ground system within 0.5 seconds of receiving the contact request.

2.1.5.4.4 After the logon function has been completed, the aircraft will send a response to the initiated ground system indicating whether or not the contact (i.e. logon) with the indicated ground system was successful. The aircraft is recommended to return the contact response to the initiating ground system within 0.5 seconds of receiving the logon response.

2.1.5.4.5 If the contact was not successful, then another contact may be attempted or another means may be employed in order to exchange the aircraft's application information with the desired ground system.

2.1.5.4.6 It is technically possible for a ground system to direct an aircraft to contact itself; i.e. the ground system sends its own facility designation and CM address in the contact request. This is in effect allowing a ground system to direct an aircraft to perform a logon with itself. While this approach may solve some of the CM addressing concerns (such as how the aircraft gets initial CM addressing information), it also introduces new concerns, such as the ground system needing to have a priori knowledge of the aircraft CM addressing information and the technical impacts of having concurrent CM connections between the aircraft and the ground system.

2.1.5.5 *Forward Function*

2.1.5.5.1 The forward function provides the capability for a ground system to forward information received from a CM logon function to another ground system. This function is initiated by a ground system, which supports ground-ground forwarding, having completed a successful logon that then forwards the aircraft CM logon information to other ground systems.

2.1.5.5.2 This allows ground systems an alternative to using the contact function for disseminating aircraft data link application information. There may be advantages to supporting this functionality: ground-ground communication may be more cost and time effective for application information dissemination than air-ground communication, the forward function can facilitate a centralised CM server, and the forward function provides a mechanism for the receiving ground system to initiate an application information exchange with an aircraft through the update function instead of waiting for the aircraft to make the initial contact. Offsetting these advantages are added implementation costs and procedural development.

2.1.5.5.3 The forward function is a one-shot exchange of information only. There is no concept of maintaining a dialogue between two ground systems for CM.

2.1.5.5.4 The initiating ground system must provide the following information for the forward function:

a) The called and calling facility designations, to be used by the dialogue service provider for addressing purposes,

b) The CM application version number, to be used for CM application compatibility purposes,

c) The information that was contained in the logon request as received from the aircraft (Aircraft Flight ID, aircraft CM address, application information and any other flight information), and

d) An indication of the Class of Communications if required.

2.1.5.5.5 All ground systems that support CM must be capable of recognizing a forward request from another ground system. A ground system is not required to implement the full functionality associated with the processing of the received information. That is, a ground system that receives forwarded information may respond that the processing of that information is not supported. Note that receiving of forwarded information is independent from the sending of forwarding information; a particular ground system implementation may not support the processing of received forwarded information while supporting the sending of forward information. This is an implementation subset, which is detailed further in 2.4.7 of this chapter.

2.1.5.5.6 Upon receipt of forwarded information, if the receiving ground system does not support the processing of the forwarded information, it will reply to the originating ground system with a "service not supported" result.

2.1.5.5.7 If the receiving ground system does support the processing of forwarded information, it must first check to see that the CM application version numbers are compatible. If they are not, the forwarded information is rejected and the originating ground system is notified that the versions are incompatible.

2.1.5.5.8 If the version numbers are compatible, then the receiving ground system will make the information contained in the forward request available for flight plan correlation and to the other applications (such as ADS, CPDLC, FIS, etc) within the receiving ground system. The receiving ground system will then indicate to the sending ground system that the forward function was completed successfully.

2.1.5.6 *Registration Function*

2.1.5.6.1 The registration function is a local implementation that provides a method for the CM users to make available application name, address and version number for each application exchanged in the logon, update or forward functions to other applications and communication systems in the aircraft or ground system.

2.1.5.6.2 In addition to the application information available, any flight information that is exchanged is also made available and is used for correlation purposes.

2.1.5.6.3 Upon registration and correlation of an aircraft's or ground system's application information, it is a local procedure to determine when specific data link services such as ADS, CPDLC or FIS will be initiated.

2.1.5.6.4 Since the registration function is a local implementation, there are no messages specified by the SARPs for this function.

2.1.6 **Inter-relationships with Other SARPs**

2.1.6.1 There is no direct interaction between then CM application and the other applications specified in the SARPs, but the addressing information exchanged by CM is required in order to make use of the other application SARPs.

2.1.6.2 The CM SARPs also make use of the Upper Layer Application (ULA) SARPs ([2]) to perform dialogue service functions required by the CM application. Since the ULA also handles addressing, the addressing information provided by CM must be made available to the ULA service provider.

2.1.7 Structure of the SARPs

2.1.7.1 All the air-ground SARPs are produced to a standard format. This has greatly helped the maintenance of document stability, commonality and presentation. The CM SARPs are no different in basic layout from all other air-ground applications SARPs.

2.1.7.2 The CM SARPs constitute the first part of Sub-volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705).

2.1.7.3 SARPs Section 2.1.1 — Introduction — gives a very brief, high level description of CM, as an application enabling the exchange of application information in order to perform data link operations. Since this overview contains no information directly related to the stipulation of specific standards, it is almost entirely written as series of informative notes.

2.1.7.4 SARPs Section 2.1.2 — General Requirements — contains information and high level requirements for the maintenance of accommodation (backward compatibility) and error processing.

2.1.7.5 SARPs Section 2.1.3 — Abstract Service — defines the abstract service interface for the CM Application. The CM Application Service Element (CM-ASE) abstract service is described from the viewpoint of the CM-air-user, the CM-ground-user and the CM-service-provider.

2.1.7.6 SARPs Section 2.1.4 — Formal Definition of Messages — describes the contents of all permissible CM messages through definition of the CM ASN.1 abstract syntax. All possible combinations of message parameters and their range of values are detailed.

2.1.7.7 SARPs Section 2.1.5 — Protocol Definition — splits up the specification of the CM protocol into three parts: sequence diagrams for the services covered by the abstract service, protocol descriptions and error handling for the CM-Air- and Ground-ASEs, and State Tables.

2.1.7.8 SARPs Section 2.1.6 — Communication Requirements — specifies the use of Packed Encoding Rules (PER) to encode/decode the ASN.1 message structure and stipulates the Dialogue Service requirements, including Quality of Service (QoS).

2.1.7.9 SARPs Section 2.1.7 — CM User Requirements — describes the requirements imposed on the CM-users concerning CM messages and interfacing with the CM-ASEs.

2.1.7.10 SARPs Section 2.1.8 — Subsetting Rules — specifies conformance requirements which all implementations of the CM protocol obey. The protocol options are tabulated, and indication is given as to whether mandatory, optional or conditional support is required to ensure conformance to the SARPs. These subsetting rules will permit applications to be tailored to suit individual ground implementations, commensurate with the underlying task, while still maintaining an acceptable level of interoperability.

2.1.8 Future Considerations

2.1.8.1 While CNS/ATM is still in operational use, any future version of data link will be capable of receiving and responding to a CM Version 1 logon request. Version negotiation for CM is dependent on this backwards compatibility. Without the backwards compatibility capability, there would be no way for future CM versions to recognize a logon attempt from an earlier CM version and respond properly (i.e. the protocol may not work properly).

2.1.9 References

[1] Context Management Application, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705) , Section 2.1

[2] Upper Layer Communications Service, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume IV of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[3] *Manual of Air Traffic Services Data Link Applications* (Doc 9694)

[4] Internet Communications Service, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume V of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[5] Introduction and System Level Requirements, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume I of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[6] Annex 10 — *Aeronautical Telecommunications*, Volume II

[7] *Location Indicators* (Doc 7910)

[8] *Designators for Aircraft Operating Agencies, Aeronautical Authorities and Services* (Doc 8585)

2.2 Overall General Functionality

2.2.1 General

2.2.1.1 The CM SARPs is defined as a single application handling the air-ground and ground-ground functionalities.

2.2.2 Topology

2.2.2.1 The CM application may function in either an end user-to- end user or client/server concept.

2.2.2.2 An end user-to-end user concept consists of a single aircraft communicating with a single ground system. Both the aircraft and the ground system constitute fully functional, independent end systems and have the necessary application and addressing capabilities. This is depicted in Figure 2.2-1. Note that the aircraft can communicate with more than one ground system independently. The ground system can also communicate with another ground system independently.

2.2.2.3 In this configuration, addressing information made available by the CM application may be passed, either air-ground or ground-ground, as needed and as supported by the various ground systems. Figure 2.2-2 illustrates the various methods of application information passing for a single aircraft's flight. The numbers in Figure 2.2-2 indicate the order in which the various exchanges happen and are explained below:

a) The aircraft performs a logon with Ground System A in order to exchange application information.

b) Since ground-ground connectivity exists between Ground System A and Ground System B, Ground System A forwards the information contained in the initial logon exchange to Ground System B.

c) Ground System B has a need to perform data link applications with the aircraft. Since Ground System B has received the necessary application information for the aircraft from Ground System A, an update can be performed with the aircraft. This provides the aircraft with all of the necessary application information from Ground System B. Data link application services can now take place between Ground System B and the aircraft.

d) Again, since ground-ground connectivity exists between Ground System B and Ground System C, Ground System B forwards the information contained in the initial logon exchange (from step 1) to Ground System C.

e) Ground System C performs an update with the aircraft in order to provide the aircraft with Ground System C's application information.

f) Since no ground-ground connectivity exists between Ground System C and Ground System D, the ground forwarding of application information is not able to take place. Therefore, the contact function is used. Ground System C instructs the aircraft to logon with Ground System D.

g) Upon receipt of the contact instruction, the aircraft performs a logon with Ground System D, so both Ground System D and the aircraft have each other's application information, and data link application services between them can commence.

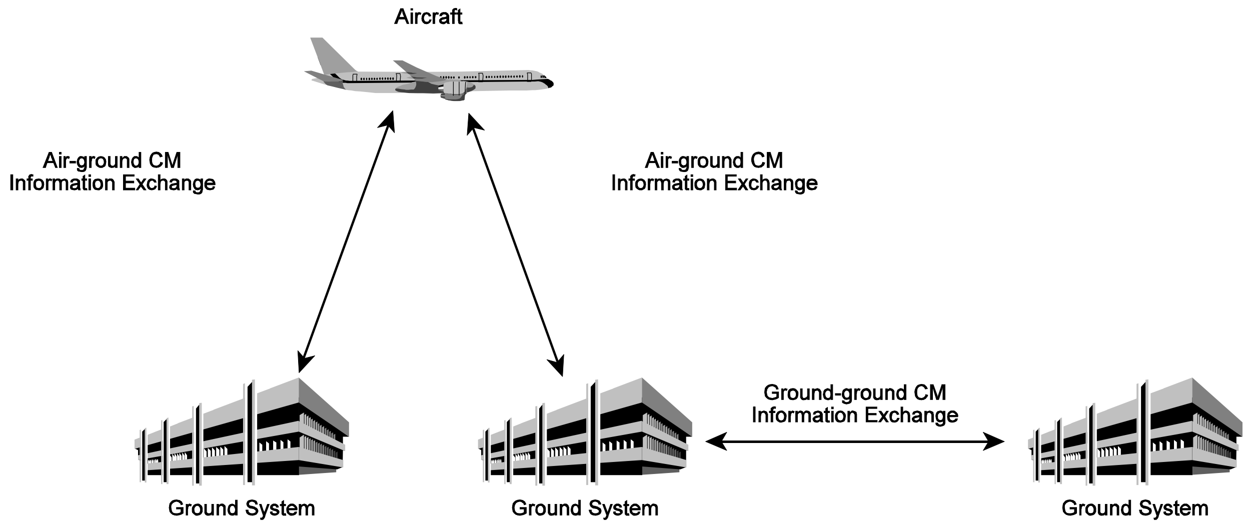

Aircraft
Air-ground CM Information Exchange
Air-ground CM Information Exchange
Ground-ground CM Information Exchange
Ground System
Ground System
Ground System

Figure 2.2-1. End user-to-end user Concept

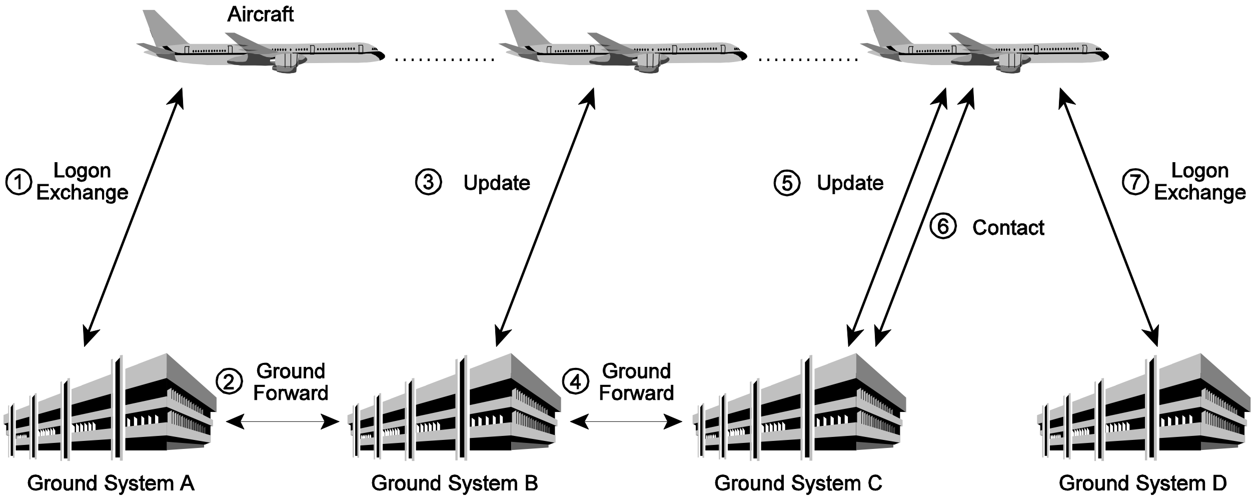


Figure 2.2-2. CM Application Information Exchanges

h) For the client/server architecture, the concept of a centralized CM server is introduced. In order for addresses of aircraft and ground systems to be readily available for a particular geographic region, a centralized CM server concept may be implemented. This is depicted in Figure 2.2-3. In this case, an aircraft will logon to a system that will store its application addresses. Other ground systems that wish to contact the aircraft can obtain the proper addressing information from the central server via a ground forwarding or other means (e.g. telephone). In that way, an aircraft is not required to logon to each facility that requires the aircraft's application information; the facility will obtain the proper information and perform a CM-update service in order to let the aircraft know what applications the facility supports. Therefore the centralized CM server may be capable of supporting communications other than Air Traffic Control (ATC).

2.2.2.4 The central CM server also provides an easy method for the aircraft to initially log on before pushback. Since the CM server is accessed by all ground systems that need the aircraft's application information, the aircraft need only ever to logon to the same CM server address. This has the advantage of being easily supported by the aircraft's avionics, and makes it very simple for initiation by the pilot (e.g. push a “logon” button) or automatically by the avionics. If the aircraft is required to contact a facility that does not make use of the central CM server, it can be told the proper CM addressing requirements through a CM-contact service by the central CM server.

2.2.2.5 Note that the links between the ground systems and the CM server as depicted in Figure 2.2-3 do not have to be CM connections. Some other communication mechanism can be used by the CM server or ground system(s) to request and/or send the appropriate

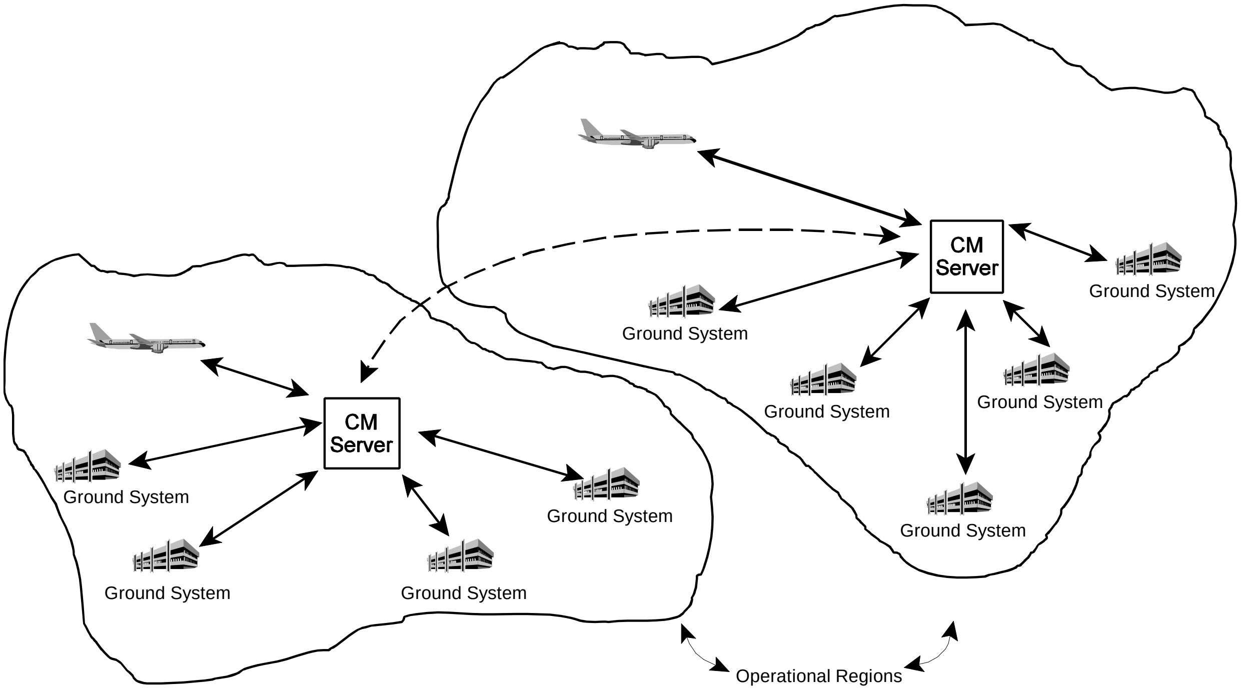


Figure 2.2-3. Centralized CM Server Concept

application information. However, it is also possible for the CM server to relay the aircraft's application information to various ground systems by using the CM forward function. The receiving ground system can then initiate the desired data link services with the aircraft.

2.2.2.6 It is also possible for two CM servers to exchange application information, as depicted by the dashed line in Figure 2.2-3. This can be done by using the forward function or by some other means. Connectivity between CM servers of particular operational regions can further simplify some of the addressing concerns for the aircraft; an aircraft may be able to logon to a CM server for the initial part of a flight and have the CM server forward the aircraft's information to an appropriate CM server of a downstream operational region.

2.2.2.7 Maintaining a CM dialogue with a centralized CM server is described in 2.2.9 of this chapter.

2.2.2.8 There are a number of considerations that need to be taken into account for a centralized CM server architecture. While a centralized CM server works well for ground-initiated applications, there are complications for applications that can be air-initiated. Unless an appropriate level of intelligence is built into the CM server, there is no way that the server can know which ground system the aircraft needs to communicate with. The only addresses that the aircraft knows is what the CM server responds with. Therefore, in order to respond with the application information of an appropriate ground system, the CM server must be able to identify from a locally stored flight plan, the flight plan information contained in the logon from the aircraft, bilateral agreements, or some other means which ground system the aircraft would need to communicate with. The CM server then must also have that ground system's application information stored locally so that it can respond to the aircraft's logon message with the appropriate application information.

2.2.2.9 The CM server also does not lend itself well if there are multiple FIS servers in an operational region. Unless the CM server has sufficient intelligence to know what FIS address the aircraft will need to use, then the aircraft would have to get the FIS application information by some other means. In the case of a single FIS server for a particular operational region, this problem is alleviated since there is only one possible FIS address to give to the aircraft. For this particular case of FIS infrastructure, a central CM server would work well.

2.2.2.10 CPDLC would also have the same problem as the multiple FIS server environment. One operational solution would be to not allow aircraft to start a CPDLC dialogue. If this is not acceptable and a CM server configuration is going to be used, then the CM server must have sufficient intelligence to know which ground system's CPDLC address to return to the aircraft. If an aircraft knows the facility designation of the ground system that it wishes to perform a CPDLC dialogue with, one option would be to include that facility designation in the logon request to the CM server. The CM server could then perform a lookup function and respond with the appropriate application information. However, this assumes that the pilot or avionics knows the facility designation that it needs to contact, and also assumes that the CM server has the functionality to perform the

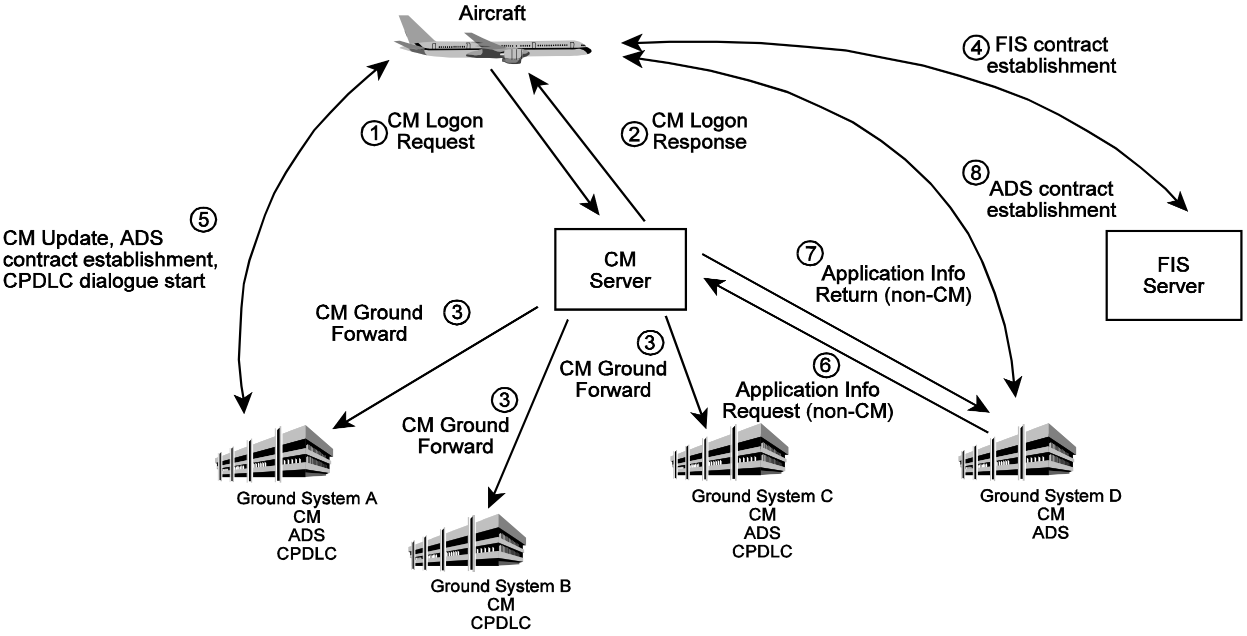

Aircraft
① CM Logon Request
② CM Logon Response
④ FIS contract establishment
⑧ ADS contract establishment
⑤ CM Update, ADS contract establishment, CPDLC dialogue start
CM Server
FIS Server
⑦ Application Info Return (non-CM)
CM Ground Forward ③
③ CM Ground Forward
③ CM Ground Forward
⑥ Application Info Request (non-CM)
Ground System A CM ADS CPDLC
Ground System B CM CPDLC
Ground System C CM ADS CPDLC
Ground System D CM ADS

Figure 2.2-4. "Dumb" CM Server Scenario

address lookup. Note that CM cannot request a single application; it must specify the facility designation where the applications reside.

2.2.2.11 Since ADS is a ground-initiated application, it does not have the same concerns as FIS and CPDLC. The CM server may return all of the ADS application version numbers that the ground systems within the operational region can support to the aircraft.

2.2.2.12 In order to further illustrate some operational scenarios with a central CM server, two examples are given; one with a "dumb" CM server (the CM server does not know enough of the aircraft's intent to determine which ground system will require data link services with the aircraft) and a "smart" CM server (the CM can determine the appropriate ground systems that require data link services with the aircraft). These scenarios are not meant to be exhaustive guidance on the implementation of a CM server, but to highlight some of the concerns of a CM server concept.

2.2.2.13 For the configuration depicted in Figure 2.2-4, the CM server does not have any knowledge of the application information of the ground systems except for CM. In this case, the CM server has the CM application address and facility designation of Ground Systems A, B and C. In addition, the CM server has ground-ground CM connectivity with Ground Systems A, B and C. The CM server does not have ground-ground CM connectivity with Ground System D. Additionally, there is a central FIS server (for simplification) which is registered with the CM server. The ground systems each support the applications as indicated. The numbers in Figure 2.2-4 are explained below:

a) An aircraft sends a CM logon request to the CM server.

b) The CM server responds with either nothing or only application name and version number for the ADS or CPDLC applications, and the FIS server's application version number and address.

c) Upon sending the logon response, the CM server automatically performs a CM Ground Forward with all ground systems with which it has ground-ground connectivity (Ground Systems A, B and C). The CM server does not know which ground systems need the aircraft's information, so it sends it to all. Ground Systems B and C have no need for the aircraft's application information at this time, so they both discard the information.

d) The aircraft desires FIS services from the server, so it establishes an FIS contract with the server. Note that if a central FIS server did not exist and instead there were several FIS locations, a ground system that performs data link services with the aircraft would be responsible for giving the aircraft the address of the appropriate FIS location by using a CM Update function.

e) Ground System A needs to have data link services with the aircraft. It also wishes to give the aircraft the option to start a CPDLC dialogue. So Ground System A sends a CM update with its CPDLC application address, and starts an ADS contract. The aircraft starts a CPDLC dialogue with Ground System A based on the information contained in the CM Update.

f) Based on locally-held flight plan information, Ground System D is expecting the aircraft to approach its sector and wishes to start ADS services. Since it does not have ground-ground CM connectivity with the CM server, it must use another means to request the aircraft's application information. Ground System D sends a request to the CM server asking for the application information for the aircraft whose Flight ID and 24 bit address is contained in the locally-held flight plan.

g) The CM server responds to Ground System D with the requested application information using a non-CM method.

h) Now that Ground System D has the aircraft's ADS application address, it establishes an ADS contract.

2.2.2.14 For the configuration depicted in Figure 2.2-5, the CM server has enough information to know what ground system an aircraft will need to perform data link services with when it receives a CM logon from that aircraft. The ground systems can depend on the server giving the proper application addressing information to the aircraft and alerting the appropriate ground system that an aircraft has performed a logon. The ground systems support the applications as indicated. The numbers in Figure 2.2-5 are explained below:

a) Aircraft 1 sends a CM logon to the CM server.

b) The CM server examines Aircraft 1's logon information, and discerns (as in 2.2.2.9 of this chapter) that it will need to communicate initially with Ground System A. The CM server retrieves the application information for Ground System A from its local database, including the FIS and CPDLC addresses and version numbers and ADS version number, and replies to Aircraft 1 with this information contained in the logon response.

c) The CM server then needs to notify Ground System A that Aircraft 1 has its application information, and also needs to make sure that Ground System A has Aircraft 1's application information. Since the CM server knows that ground-ground CM connectivity does not exist between itself and Ground System A, the CM server uses a different means to pass Aircraft 1's application information to Ground System A.

d) Upon receipt of the Aircraft 1's application information, Ground System A establishes an ADS contract. Aircraft 1 then establishes a FIS contract.

e) Aircraft 2 sends a CM logon to the CM server.

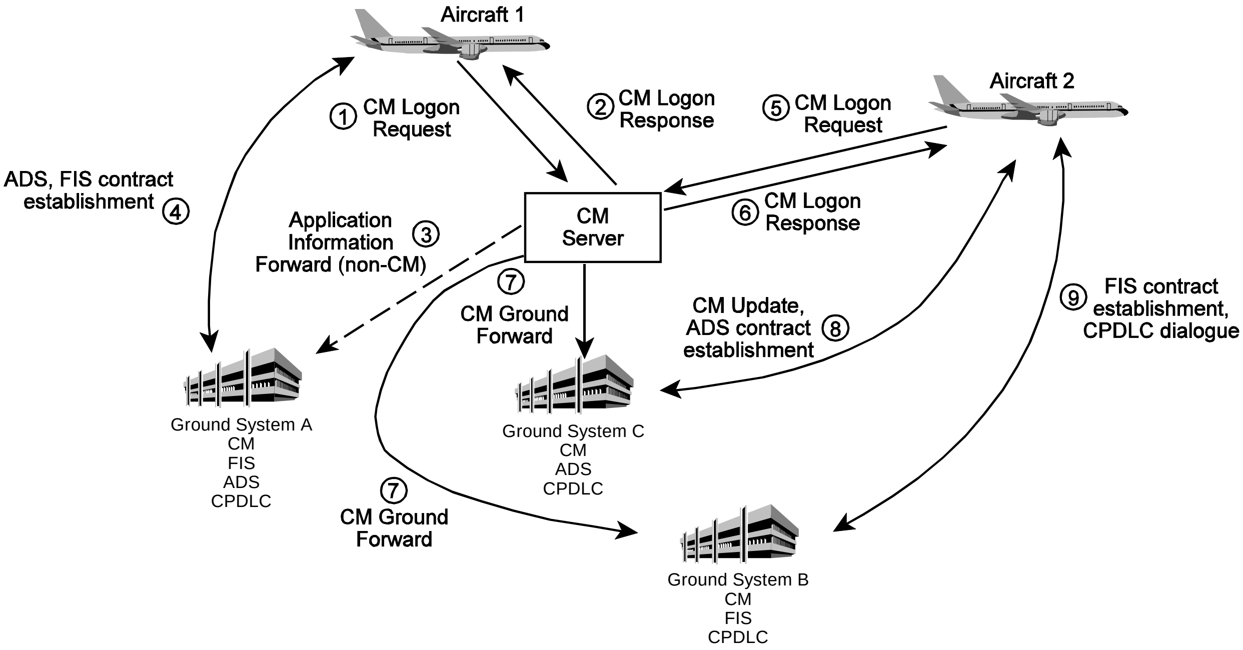


Figure 2.2-5. "Smart" CM Server Scenario

f) The server discerns (as in 2.2.2.9 of this chapter) that Ground System B will need to communicate with Aircraft 2. The CM server replies with Ground System B's application information, including FIS and CPDLC addresses and version numbers.

g) The CM server then needs to notify Ground System B that Aircraft 2 has its application information, and also needs to make sure that Ground System B has Aircraft 2's application information. Accordingly, the CM server uses the CM ground forward function to pass Aircraft 2's application information to Ground System B. In addition, the CM server also realizes (as in 2.2.2.9 of this chapter) that Ground System C will need to monitor Aircraft 2 using ADS, so the CM server uses the CM ground forward function to pass Aircraft 2's application information to Ground System C as well.

h) Ground System C decides that it wants to give Aircraft 2 the option of starting a CPDLC dialogue. Using the information forwarded to it by the CM server, Ground System C sends a CM Update to Aircraft 2 with the appropriate application information and then establishes an ADS contract.

i) Ground System B uses the information forwarded to it by the CM server to establish a CPDLC dialogue. Aircraft 2 uses the FIS information given to it in the logon response to establish a FIS contract with Ground System B.

2.2.3 Abstract Internal Architecture

2.2.3.1 *General*

2.2.3.1.1 The architectural model for the CM Application conforms to [2]. One Application Entity (AE) is defined per CM Application. Each AE contains an ATN Application Service Element (ASE), which is the communication element responsible for the ATN Application. The CM SARPs specify the CM AE and CM ASE.

2.2.3.2 *Type of ASE in CM*

2.2.3.2.1 The CM application defines only one type of ASE — the CM-ASE. The CM-ASE has two variations, the CM-air-ASE and CM-ground-ASE. Additionally, the CM-ground-ASE has three functional modes, the CM-ground-ASE used for air-ground communications, the CM-ground-ASE used for initiating ground-ground communications, and the CM-ground-ASE used for responding to ground-ground communications. The CM-air-ASE and CM-ground-ASE (air-ground) are used for air to ground communication, while the CM-ground-ASE (ground-ground initiator) and CM-ground-ASE (ground-ground responder) are used for the ground forward function. A CM-ground-ASE cannot act in two different modes at once; that is, a CM-ground-ASE cannot be an air-ground ASE and a ground-ground initiator ASE at the same time. Note, however, that a CM-ground-ASE that is a ground-ground responder is always independent of any other existing local CM-ground-ASE. The different modes of operation are depicted in Figures 2.2-6 and 2.2-7.

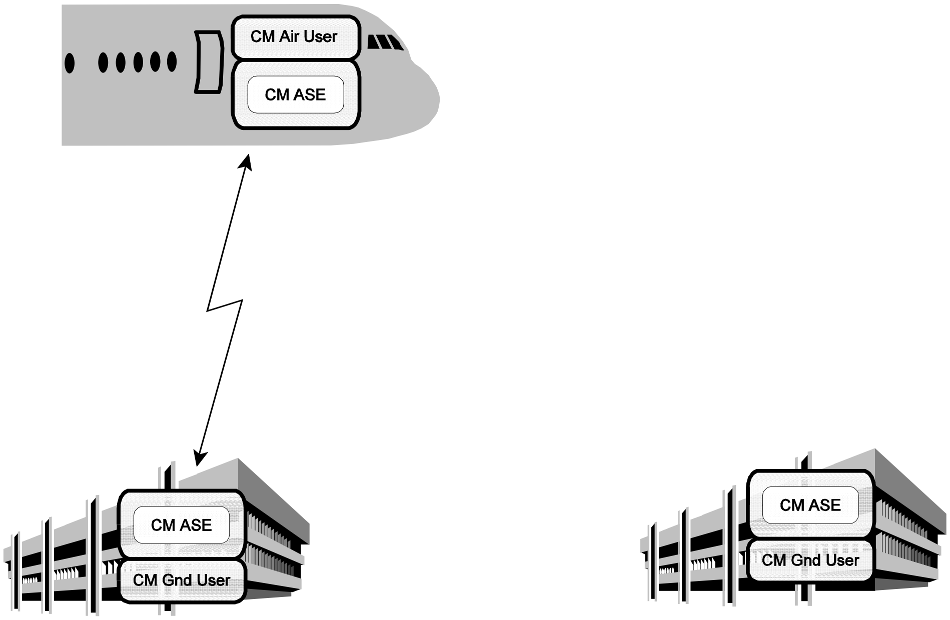


Figure 2.2-6. CM-ground-ASE in air-ground Mode

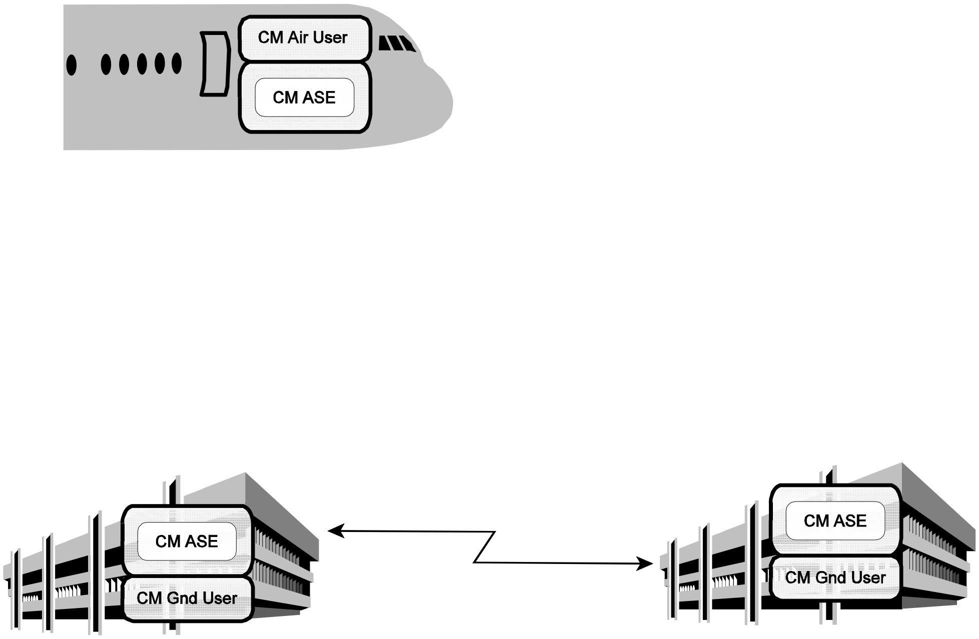


Figure 2.2-7. CM-ground-ASE in ground-ground Initiator and Responder Mode

2.2.3.3 *CM-ASE Functionality*

2.2.3.3.1 The CM-ASEs are responsible for all aspects of air-ground and ground-ground communication related to CM. They encode and decode the PDUs. They maintain a state machine that only allows PDUs to be sent at an appropriate time, and detect if the peer application sends PDUs at an inappropriate time.

2.2.3.3.2 There is no architectural capability for multiple instances of the CM-ASE within the same CM AE. This implies that the CM-ASE will generate and manage only one dialogue over the lifetime of the ASE.

2.2.3.4 *CM-User Functionality*

2.2.3.4.1 In the model of CM in Figure 2.2-6, there is a module called the CM-User. The functionality of the CM-User is described in 2.1.7 of the CM SARPs. The CM-ground-user is responsible for initiating Update and Contact functions, responding to Logon requests, initiating and responding to Forward functions, providing ground system application information, making peer application and flight plan information available to the ground system as defined by the Registration function, dialogue maintenance, and air- and ground-ground dissemination of CM information. There are very few requirements placed on the CM-ground-user. This allows the implementors a great deal of freedom in the method of implementation of this component.

2.2.3.4.2 The CM-air-user is responsible for initiating the Logon function, both on command (initially) and when directed by a Contact function, replying to other requests from a ground system, providing aircraft application information and making peer application and flight plan information available to the end user as defined by the Registration function. Again, there is a great deal of freedom allowed in the method of implementation of this component.

2.2.4 **Product Architecture**

2.2.4.1 The SARPs have defined an abstract model for the purposes of definition. That is, it has split the functionality between the ASE and the user and has defined an abstract interface between the two. It is strongly emphasized that there is no requirement on an implementor to build such an interface. If it is convenient, from an engineering perspective, to build an interface between two modules that embody the functionality of the ASE and the user, then the implementor is free to do so. However, if it is more convenient to build the system with interfaces in other places, then that is also acceptable. In testing a product to see if it conforms to the SARPs, no test can be made to test internal interfaces within the system.

2.2.4.2 The internal structure of the ATN ASE is not standardised across applications; for example, CM is defined as a single module while FIS is defined as seven.

2.2.5 Implementation Dependent Functionality

2.2.5.1 The CM SARPs specify some of the requirements for the user, but leave a lot up to the implementor. There are no requirements that state how the user interface appears, how CM interacts with the systems generating initial CM address information, how CM interacts with higher level functionality and with other applications such as ADS, CPDLC or FIS. All of this is implementation dependent.

2.2.5.2 Another key aspect of implementation dependence is how the initial ground CM address is supplied. The logon function is initiated by the aircraft. It can be initiated manually by the pilot or automatically by the avionics. Regardless of how it is obtained, the initial address of the CM application in the ground facility needs to be made available to the airborne CM application. This is necessary in order for the airborne CM application to know the ground address it must connect with. There are a number of ways the initial address may be supplied. Among them are:

a) The avionics determines, based on the aircraft's position, what ground system would need to be logged on to and will perform a database or table lookup function in order to obtain the address,

b) The pilot enters the facility designation or short form address of the ground system, and an on-board mechanism (such as a database or table lookup function) completes the address,

c) A pre-set address in the aircraft of a ground-based CM address regional or global file server,

d) A ground system (e.g. airline or service provider) supplies a locally stored address to the aircraft, and the aircraft makes it available to the dialogue service provider,

e) The pilot enters the full address as listed in an AIP or like document, and

f) The pilot enters a country code and a short (e.g. three letter) local address code which specifies an external addressing database, and the proper addresses are then determined by the external database and returned to the avionics.

2.2.5.3 Note that the above methods are not meant to be an exhaustive list, but gives some ways that the address may be supplied.

2.2.5.4 It is recognised that a standardised means should be established to provide initial identification of the CM facility designation and correlation with the corresponding presentation address.

2.2.6 Rationale for ASE/Users Split

2.2.6.1 The rationale for the split in functionality between the CM ASEs and the CM-users is as follows:

2.2.6.1.1 The ASE contains all functionality that is necessary to ensure the interoperability at the syntactic level. That is, two valid implementations of ASEs will be able to interact, passing data to each other in the correct order. They will be able to check the format of the data, ensure that it has been sent at an appropriate point in the dialogue and also ensure that the peer ASE is behaving according to the requirements in the SARPs. The ASE thus ensures interoperability.

2.2.6.1.2 The SARPs chapter 2.1.7 defines some requirements for the users. These are the minimum user requirements necessary to ensure the semantic interoperability of the two peers. Thus the user requirements explain how each CM service is interpreted and how it should operate.

2.2.6.1.3 Some care has been taken to ensure that the requirements are not over-specified. That is, they do not specify rules which are not absolutely essential to the syntactic and semantic interoperability of the CM function. This implementation dependent part can be built by different manufacturers in different ways, without affecting the interoperability between different implementations. This implementation dependent part has not been specified in the SARPs, but should be specified by individual product manufacturers or regional standards.

2.2.7 Inter-relationship with other ATN Applications

2.2.7.1 The pre-requisite for the use of any data link application is the knowledge by the ground system or aircraft that is initiating the service of the ATN name and address of the contacted system application. The CM application is the most natural way to retrieve this addressing information. A CM-logon exchange is therefore required between the aircraft and the ground CM system administrating the data link system. By passing the application information between the aircraft and the ground system and performing necessary address creation through the use of CM, the end systems are able to perform version negotiation within the end system itself. This makes efficient use of communication assets, as they are not required to perform version negotiation for each individual application. All of the application addresses obtained by CM will be made available to the local applications.

2.2.7.2 Support of CM is required if any data link applications are to be used (as per [5]). However, CM does not necessarily have to be involved in obtaining addresses or version negotiation of data link applications. If a ground system or aircraft has a priori knowledge of the application addresses and version numbers of data link applications it wishes to use in the peer system (e.g. from hard-coding the addressing data bases in the aircraft or the loading of this information at the gate before take-off), then a CM logon does not have to occur in order to use the applications. In this case the ground system or aircraft commences use of the desired data link application using the appropriate addresses. A drawback of this method of data link use is that if there is an equipment change or software update, the data link addresses may no longer be valid and data link services will not be possible. This can have many safety issues (e.g. connections to the wrong aircraft, aircraft route changes that change the address profile, etc). Also note that since the dialogue service provider handles addressing, the data link applications' address must be made available to the dialogue service provider.

2.2.7.3 The CM application is a standalone application which can be developed, certified, installed and operated completely independently from the other ATN applications.

2.2.7.4 A CPDLC application may require the address of a peer CPDLC application, such as in the case of a DSC connection. This address may be obtained through CM, although this assumes that the aircraft has knowledge of the CM address of the facility with which it wishes to establish a DSC connection (the same options for determining the CM address as listed in 2.2.5 of this chapter would also apply here). If CM is not to be used for determining the address to enable the DSC connection, then the CDPLC DSC address will have to be known by having it published in an AIP or like document or by having it sent by the CDA in a CPDLC message. There are other considerations, such as when the DSC connection becomes the CDA connection, other data link application information at the CDA location may not be known, so other data link services will not be available.

2.2.8 Distribution of CM Information

2.2.8.1 In order to support the CM operational concept as defined in [3], the CM application must be able to transfer data link application information to appropriate ATS facilities when needed. This function is achieved through the use of the CM-contact service and CM-forward service.

2.2.8.1.1 Ground-ground Distribution

2.2.8.1.1.1 Ground-ground distribution is facilitated by the CM-forward service. The CM-forward service is used to send the information received in a CM-logon message to an appropriate ATS facility. The receiving ATS facility can then use the received aircraft application information to perform a CM-update service in order to notify the aircraft of its data link application information.

2.2.8.1.1.2 The CM application does not provide a mechanism for an ATS facility to request an aircraft's application information from another ATS facility. If a ground facility wants to get an aircraft's application information forwarded to itself, it must use some other means to request the information.

2.2.8.1.2 Air-ground Distribution

2.2.8.1.2.1 Air-ground distribution can be facilitated by the use of the CM-contact service. Although this service is not an exact air-ground equivalent of the CM-forward service, it does provide a functional means to have an aircraft exchange application information to another ATS facility via a CM-logon service. This may be used in areas where ground-ground connectivity is not available, or if an ATS facility does not support the CM-forward service. Note that the aircraft may not actually perform the logon. In this case, the ATS facility that invoked the CM-contact service would get an indication from the aircraft that the contact was not successful. Another means must then be used to forward the application information.

2.2.8.2 The CM application does not provide a mechanism for an ATS facility to request an aircraft's application information from the aircraft itself. The ATS facility may use other means to instruct the aircraft to perform a CM-logon service in order to obtain the aircraft's application information (e.g. R/T).

2.2.9 Dialogue Management

2.2.9.1 *General*

2.2.9.1.1 The term “dialogue” refers to the end-to-end communication path provided by the Dialogue Service Provider. The Dialogue Service is described in detail in [2]. A dialogue is mapped directly onto a transport connection.

2.2.9.1.2 The dialogue service is used by the CM ASEs for the following purposes:

C establishment, graceful release and abort of a dialogue,

C transfer of data,

C support for quality of service and version number negotiation, and

C application naming.

2.2.9.1.3 Maintaining a dialogue refers to keeping a continual connection between the aircraft and ground system. When a CM service is requested by the CM-user, if no dialogue was previously established with the peer CM system (i.e. if no connection has already been made), a dialogue is established first, then data related to the CM service are exchanged.

2.2.9.1.4 Once a dialogue is maintained, it remains open until an end service or abort is initiated. Not maintaining a dialogue refers to opening a connection only for the duration of a single service and then closing the connection as soon as the service is complete. Not maintaining a dialogue is also referred to as a “one-shot” service.

2.2.9.1.5 Maintaining a dialogue is an optional functionality for the CM-ground-user. The CM-ground-user can decide if the implementation allows, whether or not to maintain a dialogue. The aircraft does not have the option of indicating to the ground that it wishes to maintain or not maintain a dialogue.

2.2.9.1.6 The CM service hides the use of the underlying communication service to the CM-users. However, the CM-users are aware of whether or not a dialogue is maintained and is responsible for explicitly closing a maintained dialogue.

2.2.9.2 *Optimisation of the Use of Dialogues*

2.2.9.2.1 Advantages of maintaining a dialogue are apparent where there may be many different sources of application information that need to be exchanged in a short period of time due to small FIRs. Keeping a connection open eliminates the constant setting up and tearing down of connections, which will speed transaction times.

2.2.9.2.2 In the case of a centralized CM server, maintaining a dialogue may also prove beneficial. If a dialogue is maintained between the CM server and a particular aircraft, the CM server can continually update the aircraft on new application information that it receives from ground systems using it. In this way, the aircraft can efficiently receive the new application information with a minimum of connection establishment/termination time delays. However, this will also result in additional complexity since there will need to be a ground forwarding (or some other means) of application information between the CM server and ground systems.

2.2.9.2.3 There are also advantages of not maintaining a dialogue. If there is only a need for a single exchange of application information as, for example, a logon, then there would be no need to leave a connection open and have to close it down later. In this case, not maintaining a dialogue is ideal--both aircraft and ground system receive the information they need and neither has to worry about maintaining or closing the connection.

2.2.9.3 *Optional Implementation*

2.2.9.3.1 A CM ground implementor may choose to either support or not support the capability for the ground CM application implementation to maintain a dialogue. The CM air implementation must always allow for support of maintaining a dialogue.

2.2.9.3.2 If the maintain dialogue option is implemented, the CM-ground user has the option to either maintain or not maintain a dialogue when responding to a CM-logon service from the aircraft. An aircraft implementation does not have the capability to decide whether or not to maintain the dialogue; only the ground has that capability. Maintaining a dialogue may be required depending on the operating environment. If not required for a particular operational reason, the dialogue may not be maintained since the CM-logon service may be the only CM function performed between a particular aircraft and ground system.

2.2.9.3.3 If the dialogue is maintained, then the dialogue will remain open until the CM-ground-user invokes a CM-end service or an abort is sent or received.

2.2.9.3.4 The maintain dialogue function is only a capability of the CM-logon service. The CM-update and CM-contact services have no effect on whether or not a dialogue is maintained. The CM-forward service is a “one-shot” service, and has no provision for maintaining a dialogue between two ground systems.

2.2.10 **Protocol Monitoring**

2.2.10.1 *General*

2.2.10.1.1 The CM ASE ensures that the protocol is correctly handled by the peer and can be correctly operated locally. The generation and transmission of expected responses are monitored by the peer CM ASE.

2.2.10.1.2 In the case where a serious error is detected, the dialogue in place with the peer CM system is aborted. Active CM-users are informed of this situation by a CM-provider-abort indication and are provided with a reason for the abort. These cases are described in the following sections.

2.2.10.2 *Exception Handling*

2.2.10.2.1 **CM Service Provider Timers**

2.2.10.2.1.1 If the CM-logon response is not invoked by the CM-ground-user within a period of time locally specified in the aircraft (recommended as 4 minutes in the CM SARPs), the dialogue in place with the ground CM system is aborted. CM-users are informed of this situation by a CM-provider-abort indication received at both sides with a reason of "timer-expiry".

2.2.10.2.1.2 If the CM-contact or CM-forward response is not invoked by the CM-air-user or CM-ground-user (respectively) within a period of time locally specified in the ground system (recommended as 8 minutes and 4 minutes, respectively, in the CM SARPs), the dialogue in place with the aircraft or ground CM system is aborted. CM-users are informed of this situation by a CM-provider-abort indication received at both sides with a reason of "timer-expiry".

2.2.10.2.1.3 When a dialogue is to be closed, a D-END is initiated via a CM-end service. If a D-END confirmation is not received by the CM-ground-ASE within a period of time locally specified in the ground system (recommended as 4 minutes in the CM SARPs), the dialogue in place with the aircraft is aborted. Active CM-users are informed of this situation by a CM-provider-abort indication with a reason of "timer-expiry".

2.2.10.2.2 **Unrecoverable System Error**

2.2.10.2.2.1 The unrecoverable system error is intended to cover cases where a fault causes a system lockup or the system becomes unstable (e.g. the system gets short on memory). Upon determining that it has become unstable, the CM application will attempt to abort with the reason "undefined-error".

2.2.10.2.2.2 The unrecoverable system error processing is written as a recommendation in the SARPs instead of a requirement, as it is recognized that depending on the nature of the error in the system, it may not be possible to regain control in order to perform an abort.

2.2.10.2.3 **Invalid PDU**

2.2.10.2.3.1 An invalid PDU is defined as a PDU that cannot be decoded or that is not received when it is expected. For example, a PDU that somehow becomes garbled would be an invalid PDU, or a logon request received from an aircraft that did not have a Logon Request parameter would also be considered an invalid PDU. There are separate cases for handling invalid PDUs, one for the receipt of an Indication primitive (for D-START indication and D-DATA indication) and one for receipt of a Confirmation primitive (for D-START confirmation).

2.2.10.2.3.2 For the case of the D-START indication and D-DATA indication, there is inherently a connection in place and if there is an invalid PDU then a D-ABORT must be invoked. For this case, the reason "invalid-PDU" is given in the abort. If the CM-air-user or CM-ground-user is an active user, then an indication that an abort due to an invalid PDU occurred is given.

2.2.10.2.3.3 For the case of the D-START confirmation, there may or may not be a connection in place. If there is a connection in place (i.e. a dialogue is attempting to be started), then a D-ABORT will be invoked with the abort reason "invalid-PDU". If a connection is not in place, only the local active user need be informed that an abort due to an invalid PDU occurred. Note that the active user is notified regardless of whether or not a connection is in place.

2.2.10.2.4 Not Permitted PDU

2.2.10.2.4.1 A not permitted PDU is defined as a PDU that arrives when the CM-ASE is in a state that does not allow that PDU. For example, receiving a CMContactResponse from an aircraft when the ground system is in the LOGON state. This does not mean that the PDU cannot be decoded; that situation is an invalid PDU as described above. As for the invalid PDU, there are separate cases for handling not permitted PDUs, one for the receipt of an Indication primitive (for D-START indication and D-DATA indication) and one for receipt of a Confirmation primitive (for D-START confirmation).

2.2.10.2.4.2 For the case of the D-START indication and D-DATA indication, there is inherently a connection in place and if there is an invalid PDU then a D-ABORT must be invoked. For this case, the reason "not-permitted-PDU" is given in the abort. If the CM-air-user or CM-ground-user is an active user, then an indication that an abort due to a not permitted PDU occurred is given.

2.2.10.2.4.3 For the case of the D-START confirmation, there may or may not be a connection in place. If there is a connection in place (i.e. a dialogue is attempting to be started), then a D-ABORT will be invoked with the abort reason "not-permitted-PDU". If a connection is not in place, only the local active user need be informed that an abort due to an invalid PDU occurred. Note that the active user is notified regardless of whether or not a connection is in place.

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

2.2.10.2.5.1 Only the CM-logon service provides the ability for the CM-ground-user to maintain a dialogue. This results in a D-START Confirmation received by the CM-air-ASE with the D-START Result parameter being set to the abstract value of "accepted". The CM-air-user never has the option to maintain a dialogue when responding to a CM-contact or CM-update service, nor does a CM-ground-user when using the CM-forward service. This is ensured by 2.1.5.4.5.1 in the CM SARPs, which checks D-START confirmations that are received by the CM-ground-ASE for the Result parameter being set to the abstract value of “accepted”. If this is the case, then a D-ABORT is invoked with the reason “dialogue-acceptance-not-permitted”. If the CM-ground-user is an active user then an indication that an abort due to an illegal value of the D-START Result parameter occurred is given.

2.2.10.2.5.2 In the normal course of operation, CM-ASEs should not receive a D-START confirmation with the D-START Result parameter having the abstract value “rejected (transient)” nor D-START Reject Source parameter “ DS provider”. The D-START Result parameter is set by the CM-ASEs, and is normally either “rejected (permanent)” as in the case of a dialogue not being maintained or “accepted” for the case of a dialogue being maintained. The D-START Reject Source parameter is set by the dialogue service provider, and is normally “ DS User”. The D-START Result abstract value “rejected (transient)” or the D-START Reject Source abstract value “ DS provider” is indicative of a communication failure, since they both specify that the dialogue was rejected by the provider. Therefore, an indication that an abort due to a communication service error is given to the CM-air-user or CM-ground-user, if active. A D-ABORT is not invoked since there is inherently no connection due to the communication error condition.

2.2.10.2.6 D-END Confirmation Not as Expected

2.2.10.2.6.1 When a CM-end service is requested by a CM-ground-user, a D-END request is invoked. Upon receipt of the corresponding D-END indication, the CM-air-ASE’s only choice is to accept the D-END by setting the D-END response Result parameter to the abstract value “accepted”. If for some reason this value is not present, the CM-ground-ASE will invoke a D-ABORT with the reason of “dialogue-end-not-accepted”. An indication is not given to the CM-ground-user, since the dialogue was already expected to end, regardless of whether it was graceful or not.

2.2.10.2.7 D-START Indication Quality of Service Parameter Not as Expected

2.2.10.2.7.1 The ATN Sub-volume I ([5]) dictates the values used for application service priority and RER quality of service parameters for all ATN applications. These values must be adhered to so proper levels of flight safety and performance are maintained. For CM, the values used are “flight regularity communications” for application service priority and “low” for RER quality of service. If these values are not present, then the CM-ASE will abort with reason “invalid-QOS-parameter”.

2.2.10.2.8 Expected PDU Missing

2.2.10.2.8.1 With the exception of D-END primitives, if the user data parameter of a received indication or confirmation does not contain an APDU, this means that the peer has not correctly run the CM protocol. The dialogue is aborted by the CM-ASE receiving the empty primitive by invoking a D-ABORT with the reason “expected-PDU-missing”.

2.2.10.2.9 PDU Received when Not Expected

2.2.10.2.9.1 This error is not addressed in the CM SARPs. The CM-end service is the only service that does not allow a PDU in the user data parameter. Since this service is only invoked when the dialogue is to be terminated, the presence of a PDU in the primitive would not affect the intended purpose of the service.

2.2.11 Version Number Negotiation

2.2.11.1 The CM SARPs specify the operation of version 1 of the CM application. The version number is a value inherent to the CM and is not provided by the CM users. Since the CM version number is hard coded and changes only with subsequent versions of the SARPs, there will be no confusion due to an entry error or non-standard versions.

2.2.11.2 When performing a CM-logon service, the air and ground users can discern whether or not their respective CM implementations are compatible. The version negotiation for the CM application is handled by the CM protocol. Versions numbers of other ATN applications are provided by CM, but it is up to the users of the other ATN applications to use the correct version numbers in operation. Other ATN applications do not perform individual application version negotiation.

2.2.11.3 There are three possible cases for CM version number comparisons:

a) the aircraft's CM version number is greater than the ground's,

b) the aircraft's CM version number is less than the ground's, and

c) the aircraft's CM version number is equal to the ground's.

2.2.11.4 For the case of 2.2.11.3 a, there is no guarantee of backwards compatibility from the ground user's point of view. Therefore the logon is rejected by the ground's CM-ASE and no application information is exchanged. If the aircraft is capable of supporting an earlier version that is supported by the ground system, another logon may be initiated.

2.2.11.5 For the case of 2.2.11.3 b, the CM application on the ground is assumed to be backwards compatible with the CM application on the aircraft. The application information is accepted by the ground system, and the lower-numbered aircraft's CM application version number is made available to the ground user, and the ground's CM information is returned to the aircraft. Since the exchange of CM information was acceptable to the CM-ground-ASE, both peer users are then able to perform individual ATN application version negotiation. The ground user can then decide if any subsequent action needs to be taken based on the aircraft's CM application version number (i.e. whether the ground user wants to perform CM services with an older version of CM).

2.2.11.6 For the case of 2.2.11.3 c, the CM applications in the air and ground are identical versions. There is no need to make the CM version numbers available to either user since the version numbers are identical. Therefore, the exchange of application information commences, and both peer users have the ATN application version numbers.

2.2.11.7 CM version negotiation for ground-ground exchanges using the CM-forward service are identical to those for air-ground exchanges using a CM-logon service, with the sending ground system taking place of the aircraft.

2.2.11.8 The CM-update and CM-contact services assume that CM version negotiation has already been performed.

2.2.12 CM Addressing Concerns

2.2.12.1 *General*

2.2.12.1.1 This section details some of the issues involving the use of application addresses. It gives insight into how addresses are constructed, and what address components are necessary depending on aircraft and ground system equipage, topology and implementation.

2.2.12.1.2 The CM service also provides a certain level of application address control and elementary security by limiting the knowledge of application addresses of the aircraft. Since only ATS facilities that are contacted by an aircraft, or those contacted by another ATS facility will have access to the aircraft's application addresses, only those ATS facilities will be capable of establishing application connections.

2.2.12.1.3 If new or replacement equipment or software is installed on an aircraft or within an ATS facility that affects the addressing of ATN applications, any resulting new addresses need to be made available to the CM user in order to perform data link operations. In some cases new equipment or software will have to acquire the previous addresses in order to perform data link operations.

2.2.12.1.4 The ATN is based on the 7-layer OSI reference model. The lower four layers provide the Internet Communications Service, as defined in Sub-Volume V ([4]). The upper three layers comprise the Upper Layer Communications Service, as defined in [2].

2.2.12.1.5 The ATN uses the ISO Efficiency Enhancement option for the Presentation and Session layers. These protocols in essence make the Presentation and Session layers pass through, with the overhead of only two octets (one octet for each the Session and Presentation layer). Consequently, message overhead is greatly reduced. Another result of the Efficiency Enhancement option is that the Session selector and Presentation selector are not used. This means that the normal addressing functions of these layers are not used, so there is no upper layer addressing for the ATN. Consequently, all of the addressing is accomplished with the Transport Service Access Point (TSAP) address that is defined in [4].

2.2.12.1.6 Application addresses on aircraft and within ATS facilities may not be long term. That is, the address of a particular application may be assigned as the aircraft powers up and remain in effect until the system restarts. After restart, new addresses may be assigned. This results from the assignment of the TSAP selector, which is administered on a local basis, and makes up part of an application's address. This reduces the possibility of an ATS facility storing the addresses for a particular aircraft and trying to establish an application connection at any time.

2.2.12.2 ***Address Creation***

2.2.12.2.1 In order to provide the proper addresses for applications, the TSAP address must be constructed by the CM-user. The CM SARPs mention three different types of TSAP addresses, which are conventionally termed short TSAP, long TSAP, and actual TSAP. These are defined in [4].

2.2.12.2.2 The type of TSAP used is dependent on the situation. The only types of TSAPs that are actually exchanged by the CM applications are the short and long TSAP. The short TSAP is used if the routing domain of each individual application is identical. By not sending redundant information (i.e. the routing domain information), bandwidth is saved. If the routing domain of each individual application is different, then the long TSAP (which specifies the routing domain parts) must be sent. Finally, regardless of the type of TSAP exchanged by CM, the actual TSAP must be constructed by either the CM-air-user or CM-ground-user as appropriate. Since there are parts of the actual TSAP that are identical for all ATN applications, there is no need to exchange them in CM services, and again, bandwidth is saved. However, in order to make use of the short or long TSAPs that are passed in CM services, these ATN-common parts along with the appropriate routing domain information (as appropriate) must be added by the users.

2.2.12.2.3 For the initial CM-logon service from the aircraft, the ground system's CM address needs to be known. This consists of the actual TSAP for the ground system's CM application. Possible ways of obtaining this information is detailed in 2.2.5 of this chapter.

2.2.12.2.4 Each type of TSAP is further detailed in the following sections.

2.2.12.2.4.1 **Short TSAP**

2.2.12.2.4.1.1 The short TSAP consists of the following fields:

a) the Administrative Region Selector (ARS) field, which is used to distinguish routing domains operated by the same State or Organisation,

b) the Location (LOC) field, which is used to distinguish routing areas within the same routing domain,

c) the System Identifier (SYS), which uniquely identifies the end system within the routing area,

d) the Network Service Access Point (NSAP) Selector (NSEL) which identifies the End or Intermediate System network entity or network service user process responsible for originating or receiving Network Service Data Units (NSDUs), and

e) the TSAP Selector (TSEL), which identifies the address of the access point to the transport service.

2.2.12.2.4.1.2 The ARS field is used depending on the type of network addressing domain (i.e. fixed or mobile) and whether or not there are multiple routing domains in use. In the case of an aircraft (mobile network addressing domain), the ARS is mandatory as it identifies the aircraft on which the addressed system is located. For aircraft, the ARS takes the value of the aircraft 24 bit address. If more than one routing domain exists on board an aircraft then the LOC field is used to distinguish between them. In the case of the ground (fixed network addressing domain), the ARS field is optional. It need only be used if more than one routing domain exists on the ground. For ground systems, the ARS is set locally.

2.2.12.2.4.1.3 If the routing domain part (RDP) of the CM Long TSAP (described in 2.2.12.3.4.2 of this chapter) that is provided as part of the Logon Request parameter value is the same as an individual application's RDP, then the short TSAP may be used for that application. If the RDP of the CM Long TSAP is different than an individual application's RDP, then the long TSAP must be used for that application. This means that the short TSAP may be used for aircraft applications if there is only one routing domain on the aircraft.

2.2.12.2.4.2 **Long TSAP**

2.2.12.2.4.2.1 The long TSAP consists of:

a) the RDP, which specifies the routing domain of the application address, and

b) the short TSAP as defined above.

2.2.12.2.4.2.2 The RDP is further broken into the Version (VER), Administration (ADM), and Routing Domain Format (RDF) fields. The value of the VER field will represent either the Aeronautical Industry Service Communication (AINSC) or Air Traffic Service Communication (ATSC) organization which is responsible for the identified ATN network addressing sub-domain. The ADM value will contain the corresponding three letter representation of the responsible AINSC or ATSC organization. The RDF field is always set to a known value and is present for historical purposes.

2.2.12.2.4.2.3 If the RDP information is not the same for given applications (i.e. the same organization is not responsible for the ATN network addressing sub-domains), then the RDP needs to be passed for each application. This means that the long TSAP must be used and the short TSAP cannot be used since there will be no way to differentiate the routing domains.

2.2.12.2.4.2.4 The CM long TSAP address itself must be made available to users through published documentation. This address is needed in order to start the CM process, and cannot be discerned consistently from flight plan information and previous experience.

2.2.12.2.4.3 **Actual TSAP**

2.2.12.2.4.3.1 The actual TSAP consists of:

a) the Initial Domain Part (IDP), which is a value set by ISO to denote ATN applications, and

b) the long TSAP.

2.2.12.2.4.3.2 Since the IDP is a set value determined by ISO, it will always be the same for CNS/ATM applications and hence does not need to be sent between the aircraft and ground. It is used locally by the aircraft and ground in order to determine the full ATN address of available applications. Note that in the case of a short TSAP being passed, the long TSAP must first be created by combining the RDP that is common to the applications along with the short TSAP.

2.2.12.3 ***Flight Plan Correlation***

2.2.12.3.1 There may be a requirement by certain organizations to perform flight plan correlation for every aircraft that is requesting ATC data link services. Certainly, proper correlation of an aircraft to a flight plan ensures that the intended aircraft is being communicated with. However, the capability does exist for a ground system to accept any logon request. If the ground system does not have a flight plan for the aircraft requesting a logon, or the flight plan information from the aircraft does not correlate with the ground system's flight plan information appropriate to that aircraft, the ground system may not be capable of supplying information for a requested application.

2.2.12.3.2 CM provides the means to accomplish flight plan correlation by allowing for optional supplementary flight plan information to be exchanged by the CM-logon service. If a particular region requires supplemental flight plan information in a logon request from an aircraft in order to perform proper correlation, then local procedures must be implemented that make that optional information mandatory.

2.2.12.3.3 The filed flight plan of an aircraft will conform to the ICAO standard flight plan. The 24 bit aircraft address will be derived from the appropriate flight plan field.

2.2.12.3.4 When the CM-logon service is invoked, the aircraft will pass the 24 bit aircraft address. If the receiving ATS facility has the filed flight plan and also has the flight plan data, then the aircraft sending the logon can be correlated with the correct flight plan.

2.2.12.3.5 Additional information of Flight Identifier, Departure and Destination Airports, Time and Date of Departure may be used to resolve potential multiple filed flight plans so that correlation may take place.

2.2.12.3.6 The above paragraphs point to a logical progression of flight plan correlation:

a) The aircraft is first attempted to be correlated by the 24 bit aircraft address.

b) If correlation with the 24 bit aircraft address is not possible, then the aircraft flight IDs can be examined.

c) If there are multiple flights with the same flight ID, then the departure and destination airports and/or the time and date of departure can be examined.

d) If these steps do not allow proper correlation, then other means must be employed.

2.3 Functionality of Services

2.3.1 Introduction

2.3.1.1 This section describes first the information required by the ASEs from the CM-users. Then, it provides a high-level CM overview of the data flow within the ASE expected during normal operations.

2.3.1.2 The primitives are grouped according to the services they provide: performing a logon, update, contact, forward, ending a dialogue, or aborting a dialogue.

2.3.2 Concepts

2.3.2.1 Users of the CM service are termed CM-ground-user and CM-air-user or just CM-user when it applies to both air and ground. The CM-user represents the operational part of the CM system. It is either the final end-user (e.g. a crew member or controller) or an automated system. The CM-user that initiates a CM air-ground or ground-ground service is termed the calling CM-user or initiator. The CM-user that the initiator is trying to contact is termed the called CM-user or responder.

2.3.3 CM Service Parameters

2.3.3.1 *Facility Designation*

2.3.3.1.1 The four to eight character facility designation must be provided by the CM-user for dialogue service addressing purposes as well as for indication to the peer CM-user, and is used to identify either the called or calling ground component for a particular service. The called facility designation is used for dialogue service addressing purposes and the calling facility designation is used for indication to the peer CM-user.

2.3.3.1.2 Four characters are used to identify a particular facility, such as "CYVR" for Vancouver. Up to eight characters may be used to identify a particular function within a facility; i.e. "CYVRA123" may be the address for Vancouver en-route.

2.3.3.1.3 Refer to the appropriate ICAO documents ([6], [7], [8]) for further information on 8 character facility designations.

2.3.3.2 *Aircraft Address*

2.3.3.2.1 The 24 bit aircraft address must be provided by the CM-user for dialogue service addressing purposes as well as for indication to the peer CM-user, and is used to identify either the called or calling air component for a particular service. The called 24 bit aircraft address is used for dialogue service addressing purposes and the calling 24 bit aircraft address is used for indication to the peer CM-user.

2.3.3.3 *Class Of Communication Service*

2.3.3.3.1 The CM-logon, CM-update, CM-contact and CM-forward services each has an optional parameter called "Class of Communications Service". The Class of Communications Service parameter is a means for the CM-user to indicate the required performance in terms of end-to-end transit delay. The Class of Communications Service parameter is used to implicitly indicate that the routing class is ATSC. Also note that the Class of Communication provided for CM has no relationship to the Class of Communication for other ATN applications.

2.3.3.3.2 The Class of Communications Service, if provided by the CM-air-user or CM-ground-user, is supplied to the Transport Service Provider (TSP) in the T-CONNECT SecurityLabel parameter when the connection is established. Values for these transit delays are given in 1.3.7, Note 2 of [5].

2.3.3.3.3 The Class of Communication is not guaranteed nor is a degradation of the provided class indicated to the users. It is the responsibility of the application to ensure that the transit delay is acceptable (i.e. by proper use of application timers or time stamping).

2.3.3.3.4 If a connection is in place already when the CM-update or CM-contact service is used, then this parameter is not needed. If it is provided by the CM-ground-user when a connection is in place, it is ignored.

2.3.3.3.5 The value of the Class of Communication requested by a CM-air-user or CM-ground-user for a dialogue is not transmitted nor indicated to the peer user.

2.3.3.3.6 There is no negotiation of the Class of Communication between air and ground CM-users. If the Class of Communication is not provided a default value will be used.

2.3.3.3.7 If the CM-air-user or CM-ground-user does not require a particular Class of Communication, the Class of Communication parameter does not need to be provided. In practice this means that the implementors are free to choose what to do; for example, a default or random value could be chosen, or some other means employed. If this is the case, it means that the Class of Communication is chosen by the dialogue service provider.

2.3.3.4 *Version Number*

2.3.3.4.1 The Version Number is used to indicate version number differences between peer CM ASEs. The version number is not supplied by the CM-user. This eliminates version number ambiguity and input errors.

2.3.3.4.2 If the CM Version Number of the sending CM-air-ASE or CM-ground-ASE is less than the receiving CM-ground-ASE Version Number, then the Version Number is indicated to the receiving CM-ground-user. This is to alert the receiving CM-ground-user that there may be features in later CM application versions that are not supported. Therefore, the receiving CM-ground-user must allow for these differences in order to permit proper operation of the CM application.

2.3.3.4.3 If the CM Version Number of the sending CM-air-ASE or CM-ground-ASE is greater than the receiving CM-ground-ASE Version Number, then the Version Number is provided to the sending CM-air-user or CM-ground-user in response. Since the CM versions are incompatible and further CM interaction cannot take place, there is no need for the receiving CM-ground-user to know what the sending CM ASE Version Number is. The Version Number of the receiving CM-ASE is provided to the sending CM-air-user or CM-ground-user so that a compatible CM ASE version may be tried, if available.

2.3.3.4.4 If the CM version numbers are equal, then the Version Number is not provided to either peer user. Since both CM ASE versions are fully compatible, there is no need to provide the CM version numbers to either user.

2.3.3.5 *Logon Request*

2.3.3.5.1 The Logon Request parameter is supplied by the CM-air-user and contains the application and addressing information of the aircraft along with optional flight plan information. It is used in the CM-logon service, and is indicated to the CM-ground-user when the air CM ASE version number is less than or equal to the ground CM ASE Version Number. It is used by the CM-ground-user to make the aircraft application and addressing information available to other applications within the ground system and to correlate the aircraft with the correct flight plan.

2.3.3.6 *Logon Response*

2.3.3.6.1 The Logon Response parameter is supplied by the CM-ground-user and contains the application and addressing information of the ground system. It is used in the CM-logon service, and is indicated to the CM-air-user when the air CM ASE version number is less than or equal to the ground CM ASE version number. It is used by the CM-air-user to make the ground system application and addressing information available to other applications within the aircraft.

2.3.3.7 *Update Information*

2.3.3.7.1 The Update Information parameter is supplied by the CM-ground-user and contains the application and addressing information that will be used to update the application and addressing information held by the aircraft. It is used in the CM-update service, and is always indicated to the CM-air-user. The CM-air-user replaces the ground system application and addressing information that it currently holds with the information contained in the Update Information parameter.

2.3.3.8 *Contact Request*

2.3.3.8.1 The Contact Request parameter is supplied by the CM-ground-user and contains the facility designation and address of the ground system that the CM-ground-user requests the aircraft to contact. It is used in the CM-contact service, and is always indicated to the CM-air-user. The CM-air-user uses the information contained in the Contact Request parameter to determine the facility that it needs to perform a logon to.

2.3.3.9 *Contact Response*

2.3.3.9.1 The Contact Response parameter is supplied by the CM-air-user and contains the success, or lack thereof, of the contact request. It is used in the CM-contact service, and is always indicated to the CM-ground-user. The CM-ground-user will determine if the directed contact was successful by the information in the Contact Response parameter.

2.3.3.10 *Forward Request*

2.3.3.10.1 The Forward Request parameter is supplied by the sending CM-ground-user and contains an aircraft's application and addressing information. It is used in a CM-forward service and is indicated to the receiving ground user when the sending ground CM ASE version number is less than or equal to the receiving ground CM ASE version number. It is used by the receiving CM-ground-user to make the aircraft application and addressing information available to other applications within the ground system and to correlate the aircraft with the correct flight plan.

2.3.3.11 *Maintain Dialogue*

2.3.3.11.1 The Maintain Dialogue parameter is used to indicate whether or not a connection is to remain in place after a CM-logon service is completed. It may only be provided by the CM-ground-user, and only during a CM-logon service.

2.3.3.11.2 The Maintain Dialogue parameter may only be used if the ground CM implementation supports the maintain dialogue feature.

2.3.3.11.3 The Maintain Dialogue parameter is only used by the CM-ground-user when the CM-ground-user requires a CM dialogue to remain open.

2.3.3.11.4 If the CM-ground-user wants to maintain a dialogue when responding to a logon request from an aircraft, the CM-ground-user will set the maintain dialogue parameter to indicate that the dialogue is to be maintained.

2.3.3.11.5 If the CM-ground-user does not want to maintain a dialogue, the maintain dialogue parameter will not be provided.

2.3.3.12 *Result*

2.3.3.12.1 The Result parameter is supplied by the receiving CM-ground-ASE and is used to indicate success, or lack thereof, of the forward request. It is used in the CM-forward service, and is always indicated to the sending CM-ground-user of the forward request.

2.3.3.13 *Reason*

2.3.3.13.1 The Reason parameter identifies the reason for the CM-provider-abort. The Reason parameter is not used in the CM-user-abort. It is always indicated to the CM users.

2.3.4 **CM Logon Service Specifics**

2.3.4.1 The CM-logon service provides a means for an aircraft to exchange application information with a ground system. It is a confirmed service, and is initiated only by the CM-air-user.

- C The CM-logon service request is passed to the CM-air-ASE by the CM-air-user
  - C The CM-air-ASE:
    - C Creates a Logon Request APDU based on CM-air-user provided data
    - C Invokes a D-START request to pass:
      - C The Logon Request APDU as User Data,
      - C The Called (Facility Designation) and Calling (Aircraft Address) Peer IDs as supplied by the CM-air-user,
      - C The CM-air-ASE version number as supplied by the CM-air-ASE as DS User Version, and
      - C The QoS parameters, including the Class of Communication if provided, as Quality of Service
    - C Starts the logon timer
  - C Upon receipt of the D-START indication, the CM-ground-ASE:
    - C Confirms that the D-START contains a valid Logon Request APDU,
    - C Confirms the value of the Calling Peer ID and QoS parameters,
    - C Checks the version number of the CM-air-ASE:
      - C If the CM-air-ASE version number is greater than the CM-ground-ASE's, a D-START response is invoked with the CM-ground-ASE version number as the DS User Version and "rejected (permanent)" as the Result
      - C If the CM-air-ASE version number is less than the CM-ground-ASE's, a CM-logon service indication is invoked with the following information: Calling Peer ID (Aircraft Address), CM-air-ASE version number, and User Data (Logon Request)

- If the version numbers are equal, a CM-logon service indication is invoked with the following information: Calling Peer ID (Aircraft Address) and User Data (Logon Request)

- The CM-logon service response is passed to the CM-ground-ASE by the CM-ground-user
- The CM-ground-ASE:
  - Creates a Logon Response APDU based on the CM-ground-user provided data
  - Invokes a D-START response to pass:
    - The Logon Response APDU as User Data,
    - The Maintain Dialogue parameter:
      - if provided by the CM-ground-user, the abstract value "accepted" as the Result,
      - if not provided by the CM-ground-user, the abstract value "rejected (permanent)" as the Result
- Upon receipt of the D-START confirmation, the CM-air-ASE:
- Confirms that the D-START contains a valid Logon Response APDU,
- Stops the logon timer,
- Checks the Result to see if a dialogue is to be maintained:
  - If a dialogue is not maintained (i.e. the Result has the abstract value "rejected (permanent)"), the CM-air-ASE checks the DS User Version (CM-ground-ASE version number),
  - The CM-air-ASE invokes a CM-logon service confirmation with the following data:
    - if the CM-air-ASE version number is greater than the CM-ground-ASE's, the DS User Version (CM-ground-ASE version number),
    - if the CM-air-ASE version number is equal to the CM-ground-ASE's, the User Data (Logon Response)
  - If a dialogue is maintained (i.e. the Result is "accepted"), a CM-logon service confirmation is invoked with the following data:
    - User Data (Logon Response), and
    - Result (Maintain Dialogue)

2.3.5 **CM Update Service Specifics**

2.3.5.1 The CM Update service provides a means for a CM ground user to send updated ground application and addressing information to an aircraft. It is an unconfirmed service, and is only initiated by the CM-ground-user.

- The CM-update service request is passed to the CM-ground-ASE by the CM-ground-user
  - The CM-ground-ASE checks if a dialogue is in place or not:
    - If a dialogue is not in place, the CM-ground-ASE:
      - Creates an Update Information APDU based on the CM-ground-user provided data,
      - Invokes a D-START request with the following:
        - The Update Information APDU as User Data,

- C The Called (Aircraft Address) and Calling (Facility Designation) Peer IDs as supplied by the CM-ground-user,
- C The QoS parameters, including the Class of Communication if provided, as Quality of Service, and
- C Starts the update timer

  - C If a dialogue is in place, the CM-ground-ASE:
    - C Creates an Update Information APDU based on the CM-ground-user provided data,
    - C Invokes a D-DATA request with the following:
      - C The Update Information APDU as User Data.
- C The Update Information will arrive at the CM-air-ASE in either a D-START indication (if a dialogue is not in place) or a D-DATA indication (if a dialogue is in place):
  - C If the CM-air-ASE receives a D-START indication with the User Data containing an Update Information APDU, it will:
    - C Check to make sure that the QoS parameters are correct,
    - C Invoke a CM-update service indication with the following information:
      - C The Calling Peer ID as the Facility Designation parameter value,
      - C The User Data as the Update Information parameter value, and
    - C Invoke a D-START response with the abstract value “rejected (permanent)” as the Result
  - C If the CM-air-ASE receives a D-DATA indication with the User Data containing an Update Information APDU, it will:
    - C Invoke a CM-update service indication with the following information:
      - C The User Data as the Update Information parameter value.
- C Upon receipt of a D-START confirmation, the CM-ground-ASE will:
  - C Stop the update timer,
  - C Check to see that the Result has the abstract value “rejected (permanent)” and that the Reject Source has the abstract value “DS User”.

2.3.6 CM Contact Service Specifies

2.3.6.1 The CM-contact service allows a ground system to request that an aircraft logon to another ground system. It is a confirmed service, and is only initiated by the CM-ground-user.

- C The CM-contact service request is passed to the CM-ground-ASE by the CM-ground-user
  - C The CM-ground-ASE checks if a dialogue is in place or not:
    - C If a dialogue is not in place, the CM-ground-ASE:
      - C Creates a Contact Request APDU based on the CM-ground-user provided data,
      - C Invokes a D-START request with the following:
        - C The Contact Request APDU as User Data,
        - C The Called (Aircraft Address) and Calling (Facility Designation) Peer IDs as supplied by the CM-ground-user,
        - C The QoS parameters, including the Class of Communication if provided, as Quality of Service, and

      - Starts the contact timer
    - If a dialogue is in place, the CM-ground-ASE:
      - Creates a Contact Request APDU based on the CM-ground-user provided data,
      - Invokes a D-DATA request with the following:
        - The Contact Request APDU as User Data
- The Contact Request will arrive at the CM-air-ASE in either a D-START indication (if a dialogue is not in place) or a D-DATA indication (if a dialogue is in place):
  - If the CM-air-ASE receives a D-START indication with the User Data containing a Contact Request APDU, it will:
    - Check to make sure that the QoS parameters are correct,
    - Invoke a CM-contact service indication with the following information:
      - The Calling Peer ID as the Facility Designation parameter value, and
      - The User Data as the Contact Request parameter value, and
  - If the CM-air-ASE receives a D-DATA indication with the User Data containing a Contact Request APDU, it will:
    - Invoke a CM-contact service indication with the following information:
      - The User Data as the Contact Request parameter value.
- The CM-air-user will then perform a CM-logon service with the facility designation as identified in the Contact Request parameter. Upon completion of the CM-logon service, the CM-air user will invoke a CM-contact service response
- Upon receipt of the CM-contact service response from the CM-air-user, the CM-air-ASE will check to see if a dialogue exists or not:
  - If a dialogue is not in place, the CM-air-ASE:
    - Creates a Contact Response APDU based on CM-air-user provided data,
    - Invokes a D-START response with the following information:
      - The abstract value “rejected (permanent)” as the Result parameter,
      - The Contact Response APDU as the User Data parameter
  - If a dialogue is in place, the CM-air-ASE:
    - Creates a Contact Response APDU based on CM-air-user provided data,
    - Invokes a D-DATA request with the following information:
      - The Contact Response APDU as the User Data parameter
- The CM-ground-ASE will receive either a D-START confirmation (if a dialogue is not in place) or a D-DATA indication (if a dialogue is in place) containing the results of the contact.
  - Upon receipt of a D-START confirmation, the CM-ground-ASE will:
    - Check the User Data to ensure it contains a Contact Response APDU,
    - Stop the contact timer,
    - Check to see that the Result has the abstract value “rejected (permanent)” and that the Reject Source has the abstract value “DS User”, and
    - Invoke a CM-contact service confirmation with the User Data as the Contact Response parameter value.
  - Upon receipt of a D-DATA indication, the CM-ground-ASE will:
    - Check the User Data to ensure it contains a Contact Response APDU, and
    - Invoke a CM-contact service confirmation with the User Data as the Contact Response parameter value.

2.3.7 CM Forward Service Specifics

2.3.7.1 The CM-forward service allows a ground system to forward aircraft application data to another ground system. It is a confirmed service and is always initiated by a CM-ground-user that supports the CM-forward service.

- The CM-forward service request is passed to the sending CM-ground-ASE by the sending CM-ground-user
  - The CM-ground-ASE:
    - Creates a Forward Request APDU based on sending CM-ground-user provided data
    - Invokes a D-START request to pass:
      - The Forward Request APDU as User Data,
      - The Called (Receiving Facility Designation) and Calling (Sending Facility Designation) Peer IDs as supplied by the CM-ground-user,
      - The sending CM-ground-ASE version number as supplied by the sending CM-ground-ASE as DS User Version, and
      - The QoS parameters, including the Class of Communication if provided, as Quality of Service
  - Starts the forward timer
- Upon receipt of the D-START indication, the receiving CM-ground-ASE:
  - Confirms that the D-START contains a valid Forward Request,
  - Confirms the value of the Calling Peer ID and QoS parameters,
  - Checks to see if it supports the CM-forward service:
    - If the receiving CM-ground-ASE does not support the CM-forward service, the receiving CM-ground-ASE:
      - Creates a service-not-supported APDU,
      - Invokes a D-START response with the following information:
        - The service-not-supported APDU as the User Data,
        - The abstract value “rejected (permanent)” as the Result parameter,
    - If the receiving CM-ground-ASE does support the CM-forward service, the receiving CM-ground-ASE:
      - Checks the version number of the sending CM-ground-ASE:
        - If the sending CM-ground-ASE version number is greater than the receiving CM-ground-ASE’s, the receiving CM-ground-ASE will create a incompatible-version APDU and invoke a D-START response with the following:
          - The incompatible-version APDU as User Data,
          - The receiving CM-ground-ASE version number as the DS User Version, and
          - The abstract value “rejected (permanent)” as the Result
        - If the sending CM-ground-ASE version number is less than the receiving CM-ground-ASE’s, a CM-forward service indication as well as a D-START response is invoked.
          - The CM-forward service indication contains the following:
            - Calling Peer ID (Sending Facility Designation),

            - DS User Version (sending CM-ground-ASE version number), and
            - User Data (Forward Request)
          - The receiving CM-ground-ASE creates a success APDU and then invokes a D-START response containing the following:
            - The success APDU as User Data, and
            - The abstract value “rejected (permanent)” as the Result
        - If the version numbers are equal, a CM-forward service indication as well as a D-START response is invoked.
          - The CM-forward service indication contains the following:
            - Calling Peer ID (Sending Facility Designation), and
            - User Data (Forward Request)
          - The receiving CM-ground-ASE creates a success APDU and then invokes a D-START response containing the following:
            - The success APDU as User Data, and
            - The abstract value “rejected (permanent)” as the Result
- Upon receipt of the D-START confirmation, the sending CM-ground-ASE:
  - Confirms that the D-START contains a valid Forward Response APDU and that the Result has the abstract value “rejected (permanent)” and the Reject Source has the abstract value “DS user” ,
  - Stops the forward timer,
  - Checks the User Data to see the results of the forwarding:
    - If the User Data has the abstract value “service-not-supported”, a CM-forward service confirmation is invoked with the Result parameter set to the abstract value of “service-not-supported”,
    - If the User Data has the abstract value “incompatible-version”, a CM-forward service confirmation is invoked with:
      - The Result parameter set to the abstract value of “incompatible-version”,
      - The DS User Version Number parameter set to the CM-ASE Version Number parameter, and
    - If the User Data has the abstract value “success”, a CM-forward service confirmation is invoked with the Result parameter set to the abstract value of “success”.

2.3.8 CM End Service Specifics

2.3.8.1 The CM-end service provides the capability for a ground system to terminate a CM dialogue. This service is only used if a dialogue is in place. It is an unconfirmed service, and is only initiated by a CM-ground-user.

- The CM-end service request is passed to the CM-ground-ASE by the CM-ground-user
  - The CM-ground-ASE:
    - Invokes a D-END request, and
    - starts the end timer.
- Upon receipt of a D-END indication, the CM-air-ASE:
  - Invokes a CM-end service indication, and
  - Invokes a D-END response with the D-END Result parameter set to the abstract value “accepted”

- Upon receipt of a D-END confirmation, the CM-ground-ASE:
  - Checks to see if the Result parameter has the abstract value “accepted”, and
  - Stops the end timer.

2.3.9 CM User Abort Service Specifics

2.3.9.1 The CM-user-abort service provides the capability for an aircraft or ground system to abort communication with its peer. The CM-user can invoke an abort at any time that the CM-user is aware that any CM service is in operation, After this primitive is invoked, no further primitives may be invoked for that dialogue. It is an unconfirmed service, and may be invoked at any time by either a CM-air-user or a CM-ground-user.

- The CM-user-abort request is passed to either the CM-air-ASE or CM-ground-ASE by the CM-air-user or CM-ground-user, respectively.
  - Upon receipt of a CM-user-abort request, the CM-air-ASE or CM-ground-ASE (as appropriate):
    - Checks to make sure the ASE is active,
    - Stops any timers that are set, and
    - Invokes a D-ABORT request with the D-ABORT Originator parameter set to the abstract value “user”
- Upon receipt of a D-ABORT indication, the CM-air-ASE or CM-ground-ASE (as appropriate):
  - Checks to make sure the ASE is active,
  - Stops any timers that are set, and
  - If the CM-air-user or CM-ground user (as appropriate) is an active user, the CM-air-ASE or CM-ground-ASE:
    - Invokes a CM-user-abort service indication, if the D-ABORT Originator parameter contains the abstract value “user”, or
    - Invokes a CM-provider-abort service indication with the APDU contained in the D-ABORT User Data parameter as the CM-provider-abort service Reason parameter value. Note that neither a CM-air-user nor CM-ground-user may put an abort reason in the D-ABORT.

2.3.10 CM Provider Abort Service Specifics

2.3.10.1 When the CM ASE detects an error from which it cannot recover, the CM-provider-abort service is invoked. The CM-provider-abort service provides the capability for the CM service provider to inform its users that it can no longer provide the CM service. This service is only invoked by the CM service provider. Messages in transit may be lost during this operation.

- Upon receipt of a D-P-ABORT indication, the detecting CM-air-ASE or CM-ground-ASE (as appropriate):
  - Checks to make sure the ASE is active,
  - Invokes a CM-provider-abort service indication with the CM-provider-abort Reason parameter set to the abstract value “communication-service-failure”, and
  - The detecting ASE requests D-ABORT to abort the communication with a reason value request if the dialogue is still open. The abort originator parameter is set

to "provider". The APDU is passed to the remote system by way of the upper and lower layers and emerges in the receiving ASE.

2.3.11 CM Registration Service Specifics

2.3.11.1 CM registration is used to make the addresses that are exchanged through other CM services available to other applications in the aircraft or ground system and the dialogue service provider. There are no messages exchanged for the CM registration service. This is a local implementation issue.

2.4 CM SARPs Section Description

2.4.1 SARPs Section 2.1.2: GENERAL REQUIREMENTS

2.4.1.1 *Version Number*

2.4.1.1.1 This section of the CM SARPs is included to allow the exchange of version numbers of the CM application so that version negotiation may take place and future versions of the protocol negotiated.

2.4.1.2 *Error Processing Requirements*

2.4.1.2.1 In the abstract service definition, each service has a set of parameters and the abstract syntax of those parameters specified. Thus information which is not a valid syntax is not allowed to be input.

2.4.1.2.2 In the protocol description, it is not permitted to call a service when in an inappropriate state. Thus making use of the abstract services is not permitted at these times, e.g. a contact service cannot be invoked during the LOGON state.

2.4.1.2.3 An implementation should not allow the user to take such invalid actions; however, there is no requirement to prevent an implementation from allowing this. The error processing requirements section thus says that if the implementation allows the user to enter invalid information, the system must inform the user that an entry error has occurred. In that case, the error is locally detected and the dialogue does not need to be aborted.

2.4.2 SARPs Section 2.1.3: ABSTRACT SERVICE DEFINITION

2.4.2.1 *The Concept of an Abstract Service*

2.4.2.1.1 The CM SARPs section 2.1.3 concerns the CM abstract service. The following paragraphs provide an explanation of "abstract service".

2.4.2.1.2 In order to define the CM ASE (i.e. the part of the abstract service provider that contains the protocol machine — see section 2.1.5 of the CM SARPs), it is necessary to describe its reactions to both PDUs arriving from the peer application, and the inputs from the user. The PDUs are well defined in the protocol. The actions of the user, however, are not. The SARPs do not attempt to dictate the actions of the user except where absolutely necessary. Despite this, in order to define the ASE it is necessary to have a clear definition of user actions.

2.4.2.1.3 In order to get around this conundrum, an “abstract service” is defined. An abstract service is a textual description of the interactions between the user and the ASE. These interactions are precisely defined in section 2.1.3 of the CM SARPs. Having this definition allows the ASE to be specified precisely in terms of its reactions to the arrival of PDUs and the invocation of the services by the user. This, therefore, is the reason for defining an abstract service interface.

2.4.2.1.4 The abstract service interface is defined as being an interface between the ASE and the “user-part” of the software. These are known as the CM ASE and the CM user. The CM user is not generally the human user; it is that part of the system that uses the CM ASE.

2.4.2.2 *The Concept of APIs*

2.4.2.2.1 If one was to buy a CM application, one would be buying a suite of executable code. From the code itself it is impossible to know whether or not the abstract service interface has actually been implemented. Therefore the CM SARPs do not require that the abstract service interface has to be built as a real interface. It only requires that, when one examines it from an external point of view, it behaves in the same way as if it had been built. This is the explanation of statement 2.1.3.1.1 in the CM SARPs.

2.4.2.2.2 Thus the implementors may choose to build a CM application with a real internal interface that corresponds to the abstract service interface, or they may choose not to — it is entirely up to them. However, it should be realized that there are a number of good reasons why one might not want to build a system with an interface exactly like the abstract service interface. Examples include:

C There may be a more efficient way of building the software.

C The abstract service interface does not include parameters that are needed locally, but do not affect the state machine; for example, a real interface might include an indication of which aircraft a CM logon has come from.

C It may not be easy to build the abstract service from the development tools that are being used.

C The abstract service interface does not have any programming language bindings. A real interface would require an interface defined in a particular programming language.

2.4.2.2.3 Implementation of the abstract service interface is not mandated by the CM SARPs. The requirements for CM set out by ICAO are limited in scope — they are designed only to ensure interoperability between air and ground systems, and to ensure that they meet the stated functionality requirements. The CM SARPs do not specify the nature of any internal interface within the software, nor do they specify the human interface. Individual implementations need to define their own internal interfaces to suit their own requirements.

2.4.2.2.4 In summary, an abstract service interface is defined in the SARPs in order to be able to define the ASE protocol machine. It does not have to be built in any implementation. A real implementation of the CM SARPs would normally be expected to define its own internal interfaces. The CM application abstract service consists of seven services listed in 2.1.3.2.1 of the CM SARPs.

2.4.2.3 *Functional Model of the CM Application*

2.4.2.3.1 Figure 2.4-1 shows an abstract model for the CM application. This is the same as Figure 2.1.3-1 from the CM SARPs. Just as with the abstract service, this model shows a design of the CM application, breaking it down into modules. However, there is no requirement that an implementation actually builds it this way. The figure is presented here in order to explain the terms that are used throughout the CM SARPs. It is not required that the design of an implementation follows this structure.

2.4.2.3.2 Figure 2.4-1 shows three modules:

C the CM user (which could be a CM-air-user or CM-ground-user),

C the control function, and

C the CM ASE (application service element — which could be a CM-air-ASE or CM-ground-ASE).

2.4.2.3.3 In addition, it defines the CM application entity as the control function together with the CM ASE.

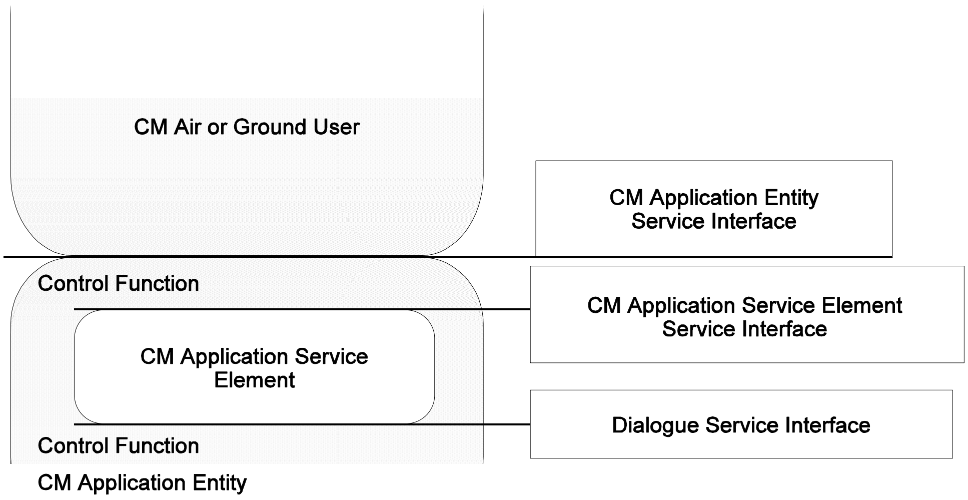

CM Air or Ground User
CM Application Entity Service Interface
Control Function
CM Application Service Element Service Interface
CM Application Service Element
Dialogue Service Interface
Control Function
CM Application Entity

Figure 2.4-1. Functional Model of the CM Application

2.4.2.3.4 Abstract interfaces are shown between the different modules:

C the CM application entity service interface — which is the same as the abstract service interface defined in 2.1.3 of the CM SARPs,

C the CM application service element service interface — which is also the same as the abstract service interface defined in 2.1.3 of the CM SARPs, and

C the dialogue service interface — which is defined in the [2], and is identical for all air-ground applications.

2.4.2.3.5 Since the CM application entity service interface is identical with the CM application service element service interface, the control function module passes calls directly from one to the other without interference.

2.4.2.4 *Conventions*

2.4.2.4.1 **Service Primitives**

2.4.2.4.1.1 The CM SARPs define seven services (CM-logon, CM-update, CM-contact, CM-end, CM-forward, CM-user-abort, and CM-provider-abort). Each primitive consists of the name of the CM service and a suffix that indicates at what point in the service the primitive occurs (request, indication, response, confirmation), e.g. CM-logon request. The primitives are further explained below:

C the CM user that initiates the service calls on the CM ASE to perform an action — this is called the "request"

C after the request is passed to the CM ASE on the other side of the communications link, it uses the service to pass the information on to its CM user — this is called the "indication"

C the CM user that has received the indication may choose to respond to it, in which case it calls upon its CM ASE to send a reply — this is called the "response"

C finally, the CM ASE receiving the response provides its CM user (which started the sequence of events) with the information — this is called the "confirmation".

2.4.2.4.1.2 The terms "request", "indication", "response" and "confirmation" are well understood in the field of communications protocols. A given CM service need not use all four primitives. Some CM services make use of one (indication), some two (request and indication or request and confirmation), some three (request, indication and confirmation) and some all four (request, indication, response and confirmation). The different cases resulting in these numbers of primitives are illustrated in Figures 2.4-2 through 2.4-6.

2.4.2.4.1.3 Each user-initiated service is also classified as being confirmed or unconfirmed. A confirmed service involves the return of an indication to the initiating user. Figures 2.4-2 — 2.4-4 are examples of a confirmed service. An unconfirmed service does not have an indication returned to the initiating user. Figures 2.4-5 and 2.4-6 illustrate a unconfirmed services.

2.4.2.4.1.4 A service is desired to be confirmed when a user requires that information be returned from the peer user. This information can be application information, such as addresses or application names, or simply a confirmation that the service was successfully completed, such as a result. It should be noted that the underlying communications service provides a means of guaranteeing notice of a services completion or non-completion.

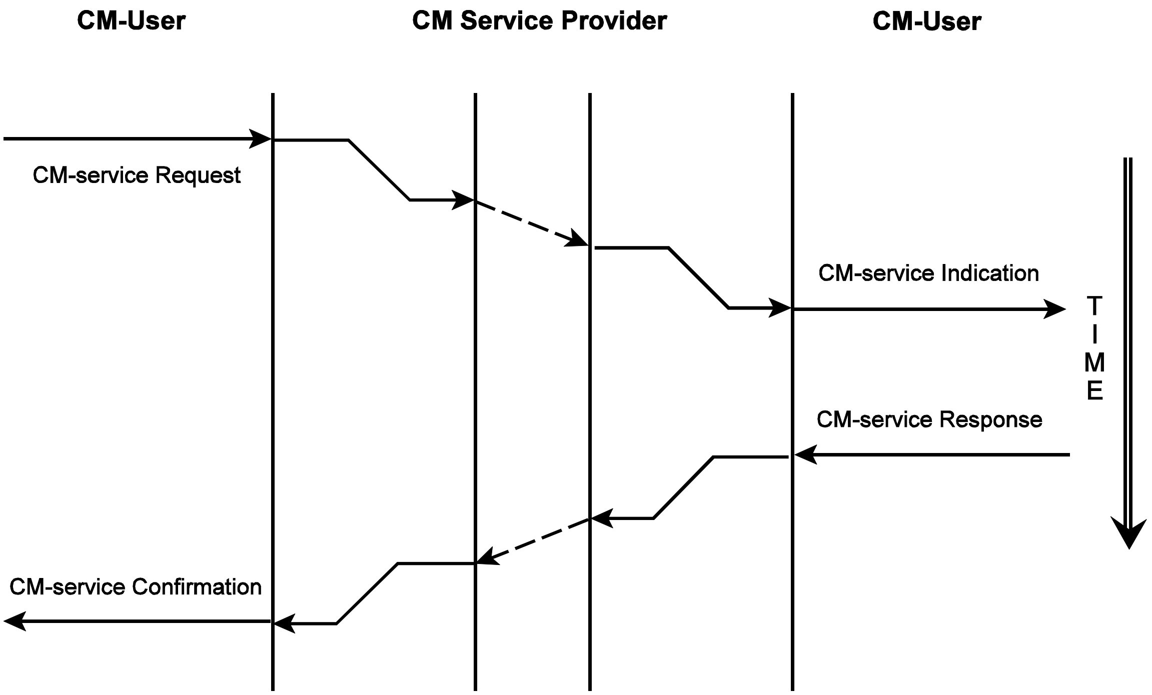


Figure 2.4-2. Generic CM-User Primitives

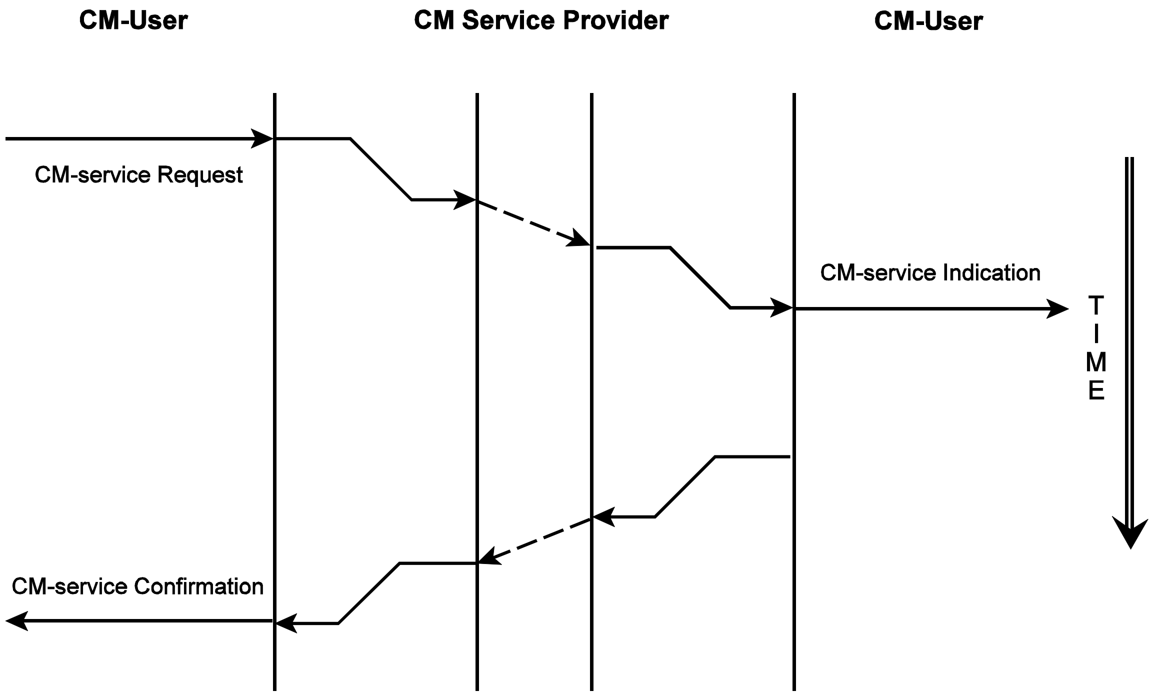


Figure 2.4-3. Example of Confirmed Service

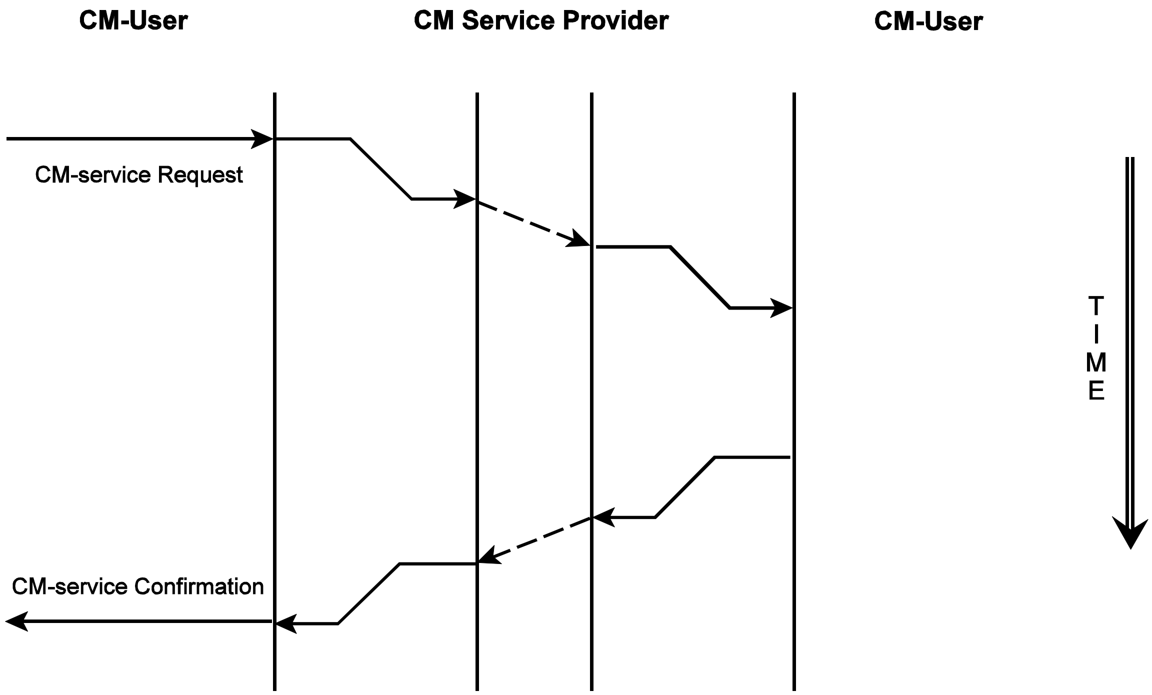


Figure 2.4-4. Example of Confirmed Service

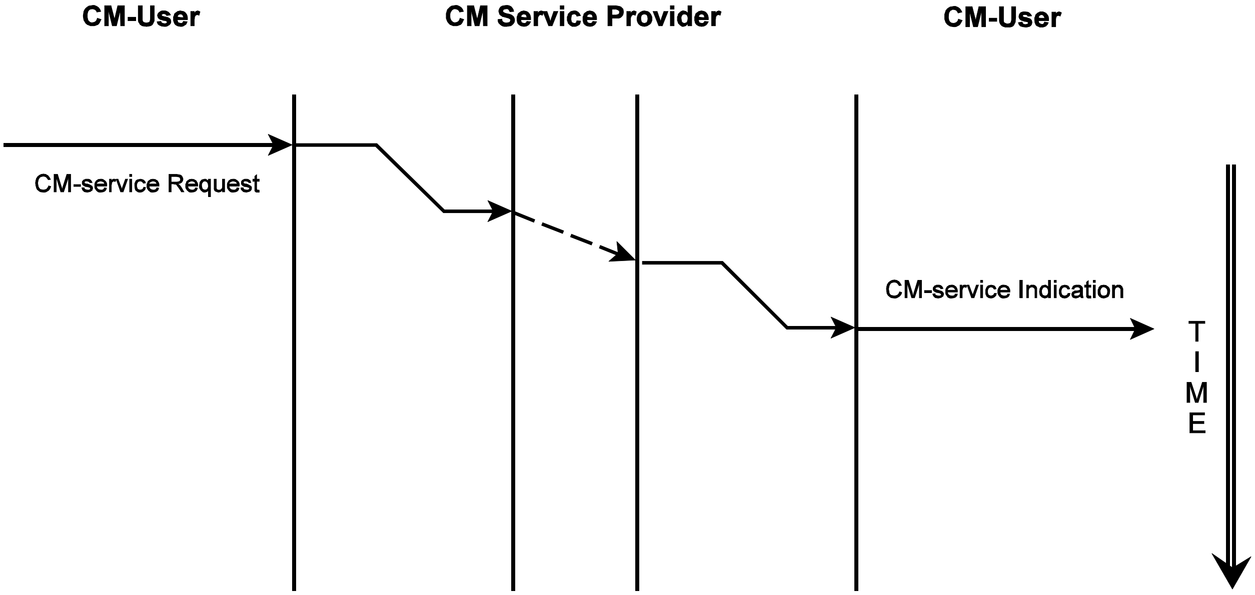


Figure 2.4-5. Generic Unconfirmed CM Service

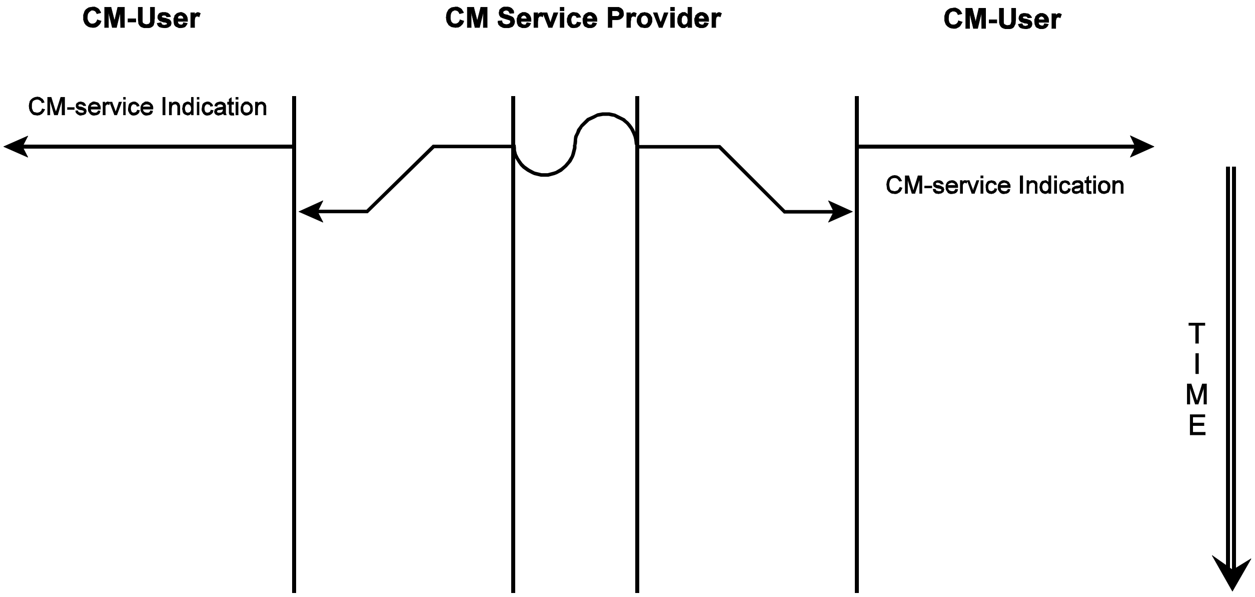


Figure 2.4-6. CM Service Provider-initiated Service

2.4.2.4.1.5 For services where application information does not need to be returned to the initiating user or when indicating a result to the initiating user would be redundant (due to the guarantee of the underlying communications service), an unconfirmed service is desirable. An unconfirmed service may save bandwidth and may be simpler to implement due to its not requiring a response.

2.4.2.4.1.6 All of the CM services in the CM SARPs take the tradeoffs between confirmed and unconfirmed services into account.

2.4.2.4.1.7 For a provider-initiated service, an indication primitive is given to both CM-users. The provider-initiated service is generated by the service provider in response to an internal condition. Figure 2.4-6 illustrates a possible provider-initiated service.

2.4.2.4.2 Detailed Service Descriptions

2.4.2.4.2.1 Each of the detailed service descriptions is defined in the same way. First there is either one or two tables indicating the parameters of the service and their status in each of the primitives. Second, each of the parameters has a short description, and a statement of what the abstract syntax of the parameter is. This is further described in the following sections.

2.4.2.4.3 Service Primitive and Parameters

2.4.2.4.3.1 Each service has a set of parameters. (one may choose to think of the service as a procedure or function calls in a programming language.) The set of parameters used in the request, indication, response and confirmation may be different.

2.4.2.4.3.2 The services are depicted in the SARPs by primitive and parameter tables. Table 2.4-1, reproduced from the CM SARPs, illustrates an example. Not all services require all primitives to be used. That is, if a particular primitive is not needed due to reasons like redundant information being relayed, then the parameter column for that primitive is omitted. If a parameter column for a primitive is present, but all of the parameters are left blank, that means that the primitive is used by the service but does not carry any user- or ASE-provided data.

Table 2.4-1. CM-logon Service Parameter Table

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

2.4.2.4.3.3 For a specific primitive, each parameter is described by a value that dictates the terms under which that parameter is used. If the use of any parameter does not follow the rules as set forth by the primitive and parameter tables, there is an error in the implementation. The abbreviations used in the primitive and parameter sections are described below:

C blank — this means that the specific parameter will not be used in this service primitive

C C — this means that the parameter is conditional upon some state. A “C” differs from a “U” (User Option) due to the fact that if the stated condition exists, the parameter must be supplied, while a “U” means that the parameter’s use is wholly up to the user. The exact conditions under which the parameter is used is explained in the text.

C C(=) — typically this is used when a request has an optional parameter. If the “C(=)” is in the “indication” position of the table, it means that if the user provides the parameter in the request, then it must also be present in the indication — what is more, it must have the same value as the parameter in the “request” column. The same applies to the “response” and “confirmation” columns.

C M — this means that the parameter must always be present, and no option not to use it exists.

C M(=) — this means that the parameter must always be present. If the parameter is in the “indication” column, then it must have a value equal to that of the “request” column. The same applies to the “response” and “confirmation” columns.

C U — this means that the use of the parameter is a user option. Therefore the presence of the parameter is completely optional, and will be used based upon user requirements. The requirements for determining the use of “U” parameters may be determined by operational or procedureal requirements outside the scope of the CM SARPs.

2.4.2.4.3.4 Where there are no parameters specified in the table (e.g. CM-end service), this means that the service has the primitives listed in the table but no parameters are needed.

2.4.2.4.3.5 The CM-logon and CM-forward services are presented with two parameter tables. The reason for this is that they may be operated in one of two ways, with each way using different primitives. The CM-logon service may be operated with a request, indication, response and confirmation, or may be operated with a request and confirmation only. The CM-forward service may be operated with a request, indication and confirmation, or may be operated with a request and confirmation only. Both cases are shown for these services.

2.4.2.4.4 **Service Parameters**

2.4.2.4.4.1 Throughout these service descriptions every parameter is described. In particular, there is a note that explains the purpose of the parameter, and a requirement that states what values it may contain.

2.4.2.4.4.2 Some primitive parameters have the same contents as APDU fields in the ASN.1 description. In most cases, the CM-ASE is tasked to copy the parameter value within the APDU field. In order to avoid confusion by defining identical data structures twice, the type of those primitive parameters is specified by simply referring to the corresponding ASN.1 type in the APDU. The ASN.1 is used in the service definition as a syntax notation only and does not implicitly imply any local encoding of these parameters. The implementation of these parameters remains a local implementation issue.

2.4.2.4.4.3 For the other parameters, the syntax is described by enumerating the authorised abstract values.

2.4.2.5 ***CM Services***

2.4.2.5.1 This section lists the CM services, and explains why each is a confirmed or an unconfirmed service.

2.4.2.5.2 The CM-logon, CM-contact and CM-forward services are all confirmed. It is necessary for the CM-logon service to be confirmed since it is the method for exchanging application information between the aircraft and the ground system. The CM-contact and CM-forward services are also confirmed since the initiating user must be made aware of the results of the requested service.

2.4.2.5.3 The CM-update, CM-end and CM-user-abort services are all unconfirmed. The CM-update is unconfirmed since there is no requirement for the ground system to know when the aircraft has received the information. The CM-end service is not confirmed since by definition the initiating ground user is inactive after the service is invoked. The CM-user-abort service is an unconfirmed service since there is no requirement for the initiating system to know that the abort has arrived at the peer (the initiating user is considered inactive after invoking the abort).

2.4.3 SARPs Section 2.1.4: FORMAL DEFINITION OF MESSAGES

2.4.3.1 ***Introduction***

2.4.3.1.1 The CM SARPs abstract syntax is written in a notation that is called ASN.1. It is strongly recommended that the reader become familiar with ASN.1 in order to understand the details of section 2.1.4.2 of the CM SARPs. ASN.1 is further explained in Part II, Chapter 7 of this manual.

2.4.3.2 *Encoding/Decoding Rules*

2.4.3.2.1 This section defines which PDUs the system must be able to encode and decode.

2.4.3.2.2 An Application Protocol Data Unit (APDU) is a sequence of bits that are passed from one peer application to another. The sequence of bits is a concrete representation of a message that is passed between applications. For example, the ASN.1 defines a message for a CMContactRequest. An APDU for this message will be a sequence of bits representing the information to be carried.

2.4.3.2.3 A system is not required to be able to encode or decode all messages specified in the ASN.1 description. An air system is not required to be able to encode a CMGroundMessage APDU nor to decode a CMAircraftMessage APDU. A ground system is not required to be able to encode a CMAircraftMessage APDU, but must be able to both encode and decode a CMGroundMessage APDU (due to the ground forwarding requirement).

2.4.3.3 *CM ASN.1 Abstract Syntax*

2.4.3.3.1 This section defines the abstract syntax of the protocol. That is, it defines the structure of the PDUs that are to be sent between aircraft and the ground systems and two ground systems.

2.4.3.3.2 Data types exchanged by CM ASEs are described in the CM SARPs by using a machine-independent and language-independent syntax. There is no constraint put on the implementors concerning the machine, nor the development language to be selected, for implementing the protocol.

2.4.3.3.3 The ASN.1 module CMMessageSetVersion1 contains the data types of the protocol data units handled by the CM ASEs. Unlike common OSI ASEs (e.g. ACSE), no object identifier has been attached to the CM ASN.1 specification. Indeed, the ULA architecture releases the applications from negotiating during the dialogue establishment as the applicable abstract syntax. Object identifiers related to CM applications (application context name and version number) are defined in the [2]

2.4.3.4 *CM ASN.1 Organization*

2.4.3.4.1 The ASN.1 itself is organised into two different sections:

C CM Message Structure

C CM Message Components

2.4.3.4.2 Both of these sections are ordered alphabetically within the sections. It should be noted that changes to the SARPs may result in the ASN.1 order losing its alphabetised order. The first elements are bolded to aid in reading.

2.4.3.4.3 The CM Message Structure is the top level choice between aircraft or ground generated messages. The CM Message Components consists of the ground and aircraft generated message components.

2.4.3.4.4 **ASN.1 Tags**

2.4.3.4.4.1 Tags are used in ASN.1 to allow to distinguish data types when confusion is possible. For instance, when a data type contains two optional elements of the same type, if only one is encoded then there is no means for the decoder to know which element the decoded value is attached to.

2.4.3.4.4.2 Even if tag values are not used by the Packed Encoding Rules, the ASN.1 grammar mandates the use of tags in some cases. When specifying the CM data types, the following rules have been used:

C tags are always used within CHOICE data type, starting at 0 and then incremented by 1 for each entry.

C tags are not used at all in SEQUENCE data type when no confusion is possible. When an optional element is defined, all elements in the sequence are tagged.

2.4.3.4.4.3 **Extensibility Markers**

2.4.3.4.4.4 In order to allow the upgrade of the ASN.1 specification when new requirements are determined, the extensibility ASN.1 feature (ellipse, depicted by "...") has been used in applicable data types. This allows future modifications to data types as the applications evolve while retaining backwards compatibility. For example, CMAbortReason has ellipses so that additional abort reasons may be added.

2.4.3.4.5 **Entry Points**

2.4.3.4.5.1 The top level (which will typically be used as an entry point in any ASN.1 compiler), is titled " Aircraft generated and Ground generated message choice". It consists of two structures: one is a choice of PDUs generated by the aircraft, and the other is a choice of PDUs generated by the ground system.

2.4.3.4.5.2 An index of the types defined in the ASN.1 is given in 2.6 of this chapter.

2.4.3.4.6 **Time Representation**

2.4.3.4.6.1 Data types have been specified for containing time indication (Date, DateTime, Year, Month, Hours, Minutes). This way of representing time is preferred over the pre-defined ASN.1 representations (GeneralizedTime and UTCTime) for optimization of the PER encoding.

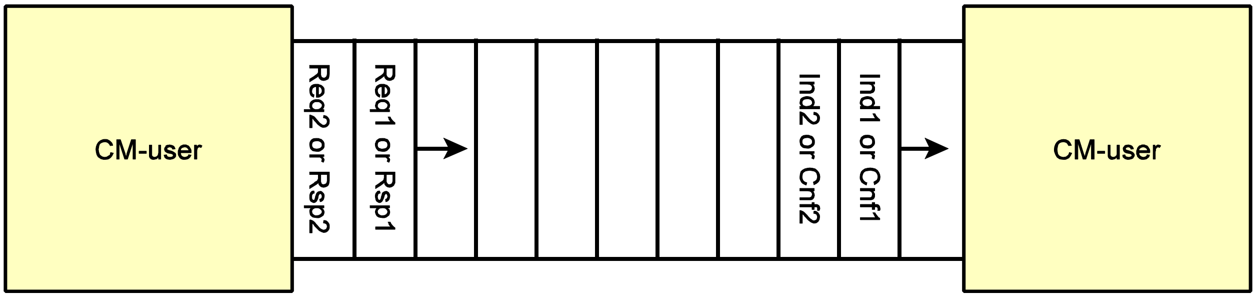


Figure 2.4-7. Notion of Delivery of Primitives in the Order Sent

2.4.4 SARPs Section 2.1.5: PROTOCOL DEFINITION

2.4.4.1 *Message Sequence Diagrams*

2.4.4.1.1 Time sequence diagrams or message sequence diagrams are used to denote the relationship between the primitives that form a CM service and the order in which they occur, e.g. the indication/confirmation primitives occur some time after the request/response primitives..

2.4.4.1.2 Inherent to the service model is the notion of queuing. The CM-service indications and confirmations are delivered to the CM-users in the order that the corresponding CM-service requests and responses were issued. This is depicted in Figure 2.4-7. One exception to the notion of queuing is the abortive services (CM-user-abort and CM-provider-abort services) which may overtake other primitives and empty the primitives in the queue. Note that for CM, Figure 2.4-7 is only relevant for the CM-Update service when a dialogue is in place.

2.4.4.1.3 Also inherent to the service model for CM is the notion that a new service may not be started while a service is still in process on the same connection. It is a local implementation issue if a CM-user will contain a queue that will accept multiple service requests, or if the user will be made to wait until a service is complete before being allowed to initiate a new service.

2.4.4.1.4 Each message sequence diagram has the same structure. A generic message sequence diagram is illustrated in Figure 2.4-8.

2.4.4.1.5 There are four vertical lines that separate the five major components in the CM system. From left to right they are:

C the CM ground user — that part of the ground system that uses the CM service to provide information to human or higher-level system users

C the CM ground ASE — that part of the ground system that implements the CM protocol

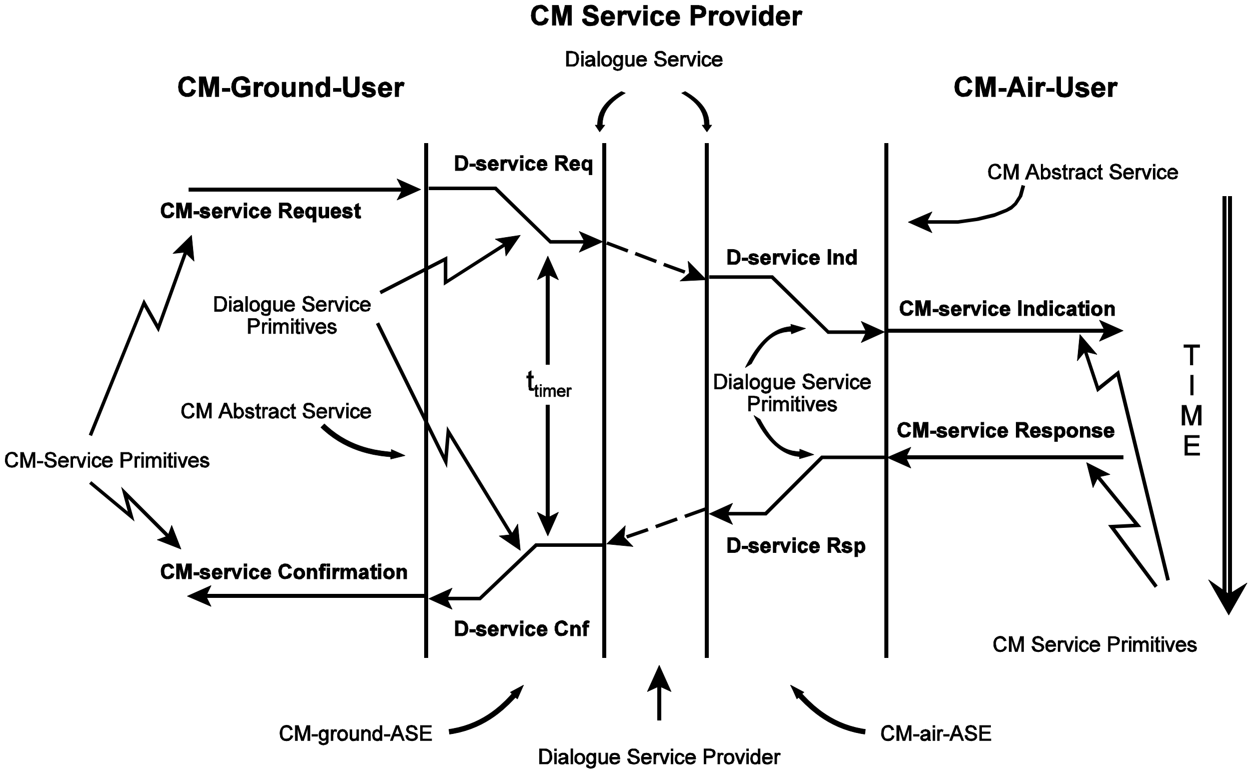


Figure 2.4-8. Generic Message Sequence Diagram

C the dialogue service provider — that part of the ground system, the air system and the networks that, together, provide the dialogue service, as defined in the upper layer architecture — on the figures this is the thin strip down the middle

C the CM air ASE — that part of the air system that implements the CM protocol

C the CM air user — that part of the avionics that uses the CM service to provide information to human or higher-level system users.

2.4.4.1.6 The middle three sections of the diagrams (CM-ground-ASE, dialogue service provider and CM-air-ASE) together form the CM service provider, and are labeled as such on the diagrams.

2.4.4.1.7 The outer two vertical lines represent the CM abstract service. Any lines crossing them represent the invocation of one of the CM service primitives. The CM service primitives are labeled in the CM-user part of the figure.

2.4.4.1.8 The inner two vertical lines represent the dialogue service. Any lines crossing them represent the invocation of one of the dialogue service primitives. The dialogue service primitives are labeled in the CM-ASE part of the figure.

2.4.4.1.9 The diagrams represent a sequence of events. Time is always considered to run down the figure from the top (representing the earliest time) to the bottom (representing the latest time).

2.4.4.1.10 If the ASEs set timers, these are marked on the figures by vertical lines with arrows at both ends. This is depicted in Figure 2.4-8 by the “ttimer” label.

2.4.4.1.11 It should be noted that the message sequence diagrams in the SARPs representing abort situations can be overlaid on top of any of the other figures to represent an abort in action.

2.4.4.2 *CM Service Provider Timers*

2.4.4.2.1 This section lists the service provider timers that are defined in the protocol, and suggests values for them.

2.4.4.2.2 The purpose of the service provider timers in the CM service provider is not operational. For operational reasons, there may be a requirement to have other timers that are shorter than the ones described here. The purpose of these service provider timers is only to ensure that the ASE protects itself when communicating with a system that has failed, for some reason, to respond.

2.4.4.2.3 Most, if not all, operational systems will require operational timers of much shorter duration than the values set for the technical timers. The value of the operational timers will also vary according to the operational scenario in which the aircraft is flying. The SARPs do not specify the value of the operational timers, nor does it insist on their implementation. For every operational system, values of operational timers will need to be calculated. Also, the action that must be taken if the operational timer expires will need to be specified. An example of such an action is: invoke a CM-user abort if the logon service or contact service is not completed in the required time and re-invoke the service with a different Class of Communication Service.

2.4.4.2.4 For example, suppose a ground system sends a contact request to an aircraft. Suppose further that the aircraft system has not implemented the CM protocol correctly, and it locks up without sending the contact response back to the ground system. The ground system will be in a state that is waiting for a result. The specification prevents the ground system from sending up another message until the first has been dealt with. This combination of events would make the ground system lock up unless a service provider timer is used. When in this state, the tcontact service provider timer will eventually reach its maximum value. The ground system can then abort the connection. Thus the ground system protects itself from getting indefinitely locked up because of another system's failure.

2.4.4.2.5 It should be noted that the values set in these service provider timers have been calculated thus: It has been assumed that the slowest possible network is in operation and that all systems in the path have their maximum delay. Should a reply to a request take longer than this, then it is assumed that something must be failing somewhere.

2.4.4.2.6 The assignment of values for timers must be optimized based on operational testing of the application. In such testing, incompatible timer values and optimum combinations can be identified. Implementations of CM protocol are required to support configurable values for all timers and protocol parameters, rather than having fixed values. This allows modification as operational experience is gained.

2.4.4.3 *CM-ASE Protocol Description*

2.4.4.3.1 The protocol description explains the rules by which the ASEs work. There is a detailed specification of actions taken by the ASEs when triggered by certain events:

C the arrival of a PDU through the dialogue service,

C the invocation of one of the service primitives by the user,

C the expiration of one of the internal timers, and

C an unrecoverable system error.

2.4.4.3.2 Both the air and ground ASEs have modules that mirror each other. The exception to this case is when the ground is in a ground-ground initiator or responder mode. In this case, the two ground ASEs mirror each other.

2.4.4.3.3 There is a requirement that the ASE does not accept the invocation of primitives when no actions are described for that primitive in that state (2.1.5.3.1.1 of the CM SARPs). Some explanation of this statement is needed. The air and ground CM ASEs have several different states. When an ASE is in a particular state, only some primitives are sensible to invoke. For example, if a CM-logon service request is invoked, it is not sensible to invoke a second CM-logon service request until a reply has been received for the first one. There is no statement in the description of the protocol that explains what the air ASE should do if it receives a second CM-logon service request before the reply to the first one has been received. The SARPs therefore require (by statement 2.1.5.3.1.1 of the CM SARPs), that the air user must not invoke a CM-logon service request during this period.

2.4.4.3.4 Thus only actions which are permitted are described. If an action is not described, then it is not permitted.

2.4.4.4 *CM-ASE State Tables*

2.4.4.4.1 State tables are provided. These should be an exact reflection of the CM protocol description, in a condensed form. The state tables are only presented for guidance, since the textual protocol description always takes precedence.

2.4.5 SARPs Section 2.1.6: COMMUNICATIONS REQUIREMENTS

2.4.5.1 *Encoding Rules*

2.4.5.1.1 The CM SARPs section 2.1.6.1 states that PER (Packed encoding rules) must be used to encode the PDUs. PER is an ISO standard and is particularly efficient at encoding data. Implementors may use ASN.1 compilers to generate code that creates PER automatically.

2.4.5.2 *Dialogue Service Requirements*

2.4.5.2.1 The dialogue service requires a number of parameters to operate. The CM SARPs section 2.1.6.1 defines those parameters that are not defined elsewhere, and states that the dialogue service must exhibit consistent behaviour with the Upper Layer Communications as defined in [2].

2.4.5.2.2 In addition to the Class of Communication Service parameter provided by the service initiator, other Quality of Service (QoS) parameters are attached by the ASE to the dialogue supporting the communication. Values for the Application Priority and the Residual Error Rate are constant for the CM application and therefore are not presented to the users.

2.4.5.2.3 The Application Priority is set by default to the abstract value “flight regularity communications”.

2.4.5.2.4 The Residual Error Rate is set by default to the abstract value of “low”.

2.4.5.2.5 The underlying layers provide a checksum and realise the requirement for message integrity. It is not necessary, therefore, to provide application level integrity by means such as redundancy checks and confirmation of services..

2.4.6 SARPs Section 2.1.7: CM USER REQUIREMENTS

2.4.6.1 The requirements set out in this section are in line with those specified by [3]. These requirements are necessary for proper operational interoperability of ATN applications, and go into more technical detail than those specified by [3].

2.4.6.2 This SARPs section also recommends values for some operational response times which are in line with the values specified for the technical timers.

2.4.6.3 The CM-user has the capability to initiate an abort at any time. However, the CM-user is not allowed to insert any data indicating a reason for the abort into the abort primitive.

2.4.7 SARPs Section 2.1.8: SUBSETTING RULES

2.4.7.1 *General*

2.4.7.1.1 There is some functionality within the CM SARPs that ground implementors may choose not to incorporate. For example, if a particular implementation of a ground system is designed never to use the contact function because it will always have ground-ground communications with other CM ground systems, then there is no requirement for it to have the code to support that function. An aircraft, on the other hand, must have the code for accepting and acting on a contact message, since it may fly into an area that does use the contact function.

2.4.7.2 There are some combinations of functionality that allow interoperability and some that do not. The CM SARPs section 2.1.8 defines the combinations of functionality that are interoperable. It should be noted that aircraft must retain full technical functionality to be SARPs compliant.

2.4.7.3 The combinations or functionality are defined in a set of tables. The purposes of these tables are summarized below:

C Version number — only one version is defined. This is a placeholder for when future versions are defined.

C Protocol Options — this defines a number of options for parts of the protocol that may be implemented. The options may be implemented together. Each has a name associated with it — the predicate.

C CM-ground-ASE configurations — this defines 28 combinations of protocol options, each of which yields a coherent protocol.

C CM-air-ASE configurations — this defines a single combination of protocol options, which is the only combination that yields a coherent protocol.

C Supported CM service primitives — this defines the conditions under which the service primitives are applicable.

C Supported CM APDUs — this defines the conditions under which the PDUs are applicable.

2.4.7.4 The subsetting rules define those subsets that are technically possible. The SARPs do not address the operational acceptability of these subsets.

2.4.7.5 Only version 1 of the CM protocol is defined. Any CNS/ATM-1 compliant CM system must support this version.

2.4.7.6 Any ground system can independently select a subset of functionality and be interoperable with any aircraft or other ground system.

2.4.7.7 *Mandatory Functionality*

2.4.7.7.1 The CM ground implementation must support, as a minimum, the logon, forward response and abort functions. This means that a minimum CM ground implementation must be capable of recognizing and responding to a logon request from an aircraft, recognizing a ground forward request from a ground system and respond with a message indicating that the use of forwarded logon information is not supported, and generating and properly reacting to all aborts.

2.4.7.7.2 A CM aircraft implementation must support all functionality specified for the aircraft in the SARPs; i.e. logon invocation, contact and update receipt, maintaining a dialogue and abort processing. This is the only valid set of functionality for an aircraft. This does not imply that the system is not SARPs compliant should part of the functionality be temporarily unavailable.

2.4.7.8 *Optional Functionality*

2.4.7.8.1 Some of the CM-services have options within itself. That is, there are more levels of functionality within a service besides full support or no support of that service. These services include the CM-logon, CM-update, CM-contact and CM-forward service.

2.4.7.8.2 The CM-logon, CM-update and CM-forward service can each have two possible implementation options for the CM-ground-ASE: one supports the maintaining of dialogue, the other does not. Since maintaining a dialogue is a CM-ground-ASE option, an implementation may not have a need to incorporate that function. Therefore, this allows an implementor to build a system that does not support maintaining a dialogue and is SARPs compliant.

2.4.7.8.3 The CM-forward service has three different options: forward initiator, forward responder and forward user. Forward initiator is used to denote a ground system that has built in the optional functionality to invoke the CM-forward service. Forward responder is used to denote the mandatory functionality that every CM ground system must employ, which is the ability to recognize a CM ground forwarded message and respond with (at minimum) “service not supported”. The forward user is used to denote a ground system that has built in the optional functionality to make use of (i.e. distribute internally within the local receiving ground system) the information that is received from a CM ground forwarded message. A ground system may choose to implement a forward initiator but not a forward user, or vice versa.

2.5 **Dimensions**

2.5.1 **PDU Size**

2.5.1.1 *Theoretical Limits*

2.5.1.1.1 Theoretically, the aircraft could invoke a logon request with all optional types of information, including application information for all 256 possible air and ground applications. The largest PDU that would have to be produced is dependent on the number of applications that need to have information provided. This is approximately 5937 octets.

2.5.1.1.2 The size of a TPDU is the size of the APDU plus the overhead introduced by transport, session, presentation and ACSE. The upper layer architecture guidance material gives an indication of the overhead in the session, presentation and ACSE (it cannot be precise, as the overhead varies depending on circumstances). Typically, they will add 2 or 3 octets onto the APDU size for a D-DATA call and 10 to 15 octets for a D-START call. The overhead introduced by the transport (e.g. the integrity check and sequencing) is not documented.

2.5.1.1.3 A factor to take into account would be the bandwidth of the air-ground subnetwork. Current technology only provides limited bandwidth across the air-ground subnetwork, so large amounts of data transmitted could monopolize the subnetwork. The size of aircraft and ground initiated PDUs are very small by comparison, and all systems should be built with the capability to receive all requests from the peer systems.

2.5.1.1.4 A second limiting factor is likely to be the memory capacity of the airborne system. Many implementations will be built into existing equipment which have limited memory available for storage of the data structures used to build and examine PDUs. Such implementations may have to recognise when they cannot cope with a request because of memory limitations, although for CM this should not present a problem.

2.5.1.1.5 Table 2.5-1 depicts typical PDU sizes for uplinks, downlinks and ground-ground exchanges.

Table 2.5-1. Typical PDU Sizes

Exchange Type	Message Type	Typical Size (Octets)	Comments
Downlink	CMLogonRequest	71	Contains 2 ground initiated application (e.g. ADS, CPDLC)
	CMContactResponse	1	
Uplink	CMLogonResponse	23	Contains 1 air initiated application, 1 ground initiated application (e.g. ADS, CPDLC)
	CMUpdate	23	Contains 1 air initiated application, 1 ground initiated application (e.g. ADS, CPDLC)
	CMContactRequest	27	
Ground-ground	CMForwardRequest	71	Contains 2 ground initiated application (e.g. ADS, CPDLC)
	CMForwardResponse	1	

2.5.1.2 *Error Handling*

2.5.1.2.1 Should either airborne or ground system receive a PDU that is too large for it to manage under its current circumstances (e.g. lack of memory due to other applications), it will be unable to decode it. The system should abort the connection under these circumstances.

2.5.1.3 *Number of CM Addresses Supported*

2.5.1.3.1 **Aircraft**

2.5.1.3.1.1 The number of CM addresses an aircraft CM implementation should be capable of storing is dependent on how the CM application is intended to be used as well as the input mechanism for the address.

2.5.1.3.1.2 A minimal implementation may provide space for only a single address. This address will be used for the initial CM logon, and will be either re-input by the pilot or avionics for a subsequent logon or replaced by the address contained in a CM contact.

2.5.1.3.1.3 It may be advisable to allow for storage of at least two CM addresses. An operational reason for this would be if a contact is received, the CM address replaced on the aircraft, and an unsuccessful logon to the new CM address is made. If the contact-originating ground-user's CM application goes down, the aircraft will then only have the non-functioning CM address. The ground system will not be able to perform interaction with the aircraft. While this is not as serious as losing all of the other application addresses of the ground system, it may cause operational problems. Therefore having at least two CM addresses is recommended.

2.5.1.3.1.4 An aircraft implementation may also wish to have storage for a list of most often used CM addresses. This will alleviate some of the address input concerns; however, the addresses may change and may not be valid.

2.5.1.3.2 **Ground**

2.5.1.3.2.1 A CM ground implementation will need to allow for as many CM addresses to be supported as perceived traffic loading requirements will mandate. That is, if the aircraft's CM address (as contained in the logon request) needs to be retained for an operational need (e.g. subsequent updates or contacts), then the number of equipped aircraft likely to be in the area will need to be calculated. This will define the number of CM addresses that should be supported.

2.5.1.4 *Number of Other Application Information Supported*

2.5.1.4.1 **Aircraft**

2.5.1.4.1.1 An aircraft will need to support at least two sets of application information for each application (other than CM) that is supported. This is due to the concern that if only one set of application information is supported, the application information obtained through a logon by command of a contact (which replaces the current ground application information in the aircraft) may prove invalid. This may leave an aircraft with no useable data link applications. CPDLC is a special case, and a subsequent allowance may need to be provided for additional CPDLC addresses to perform DSC functions. This is discussed in the CPDLC guidance material.

2.5.1.4.1.2 The maximum number of applications that an aircraft would possibly need to store would be 256. It is likely that this number of applications will not be reached for quite some time. Therefore, it is reasonable to keep the maximum number of supported applications lower. One of the near-term driving factors behind the maximum number will be the addition of ATIS applications. This needs to be taken into account when determining the maximum number of applications to support.

2.5.1.4.2 **Ground**

2.5.1.4.2.1 A CM ground implementation will need to allow for as many sets of application information to be supported as perceived traffic loading requirements will mandate. This will define the number of CM addresses that should be supported.

2.5.1.5 *Number of Concurrent CM Connections*

2.5.1.5.1 Based on peak traffic loads for data link equipped aircraft operating in the airspace in question along with whether or not there is a requirement for dialogue maintenance. The numbers specified by the ADSP can be found in [3].

2.5.1.5.2 There is nothing in the CM SARPs that restricts or dictates the number of connections that should be supported, either in the air or on the ground. This decision will be made based on many factors, including the airspace type, aircraft equipage expectations and operational procedures put into practice. Some concerns are given below.

2.5.1.6 *Aircraft CM Connections*

2.5.1.6.1 An aircraft will need to support at least two active CM connections. This is due to the contact function, where one connection is open from the ground system requesting the aircraft to contact another ground system, and the other connection is open from the ground system that the aircraft is performing the logon with. Note that both ground systems may require the dialogue to be maintained, so the aircraft will need to be capable of maintaining both dialogues. The ground system topology could in part drive the aircraft equipage. If an aircraft is operating in an environment where a CM server exists or where a dialogue is maintained to a single centre, then two connections may be sufficient operationally. However, if the aircraft is operating in an environment where there are many different centres that need to perform CM services with the aircraft concurrently, then the aircraft may need to support multiple CM connections. This will have to be determined on an operational basis. However, it is not expected that there will be very many operational scenarios that will require an aircraft to maintain a large number of CM connections. This situation is different for ground CM implementations, as is detailed below.

2.5.1.6.2 An aircraft CM implementation should take into account the possibility of receiving multiple connection requests at the same time. There is nothing in the CM SARPs that prohibit a CM implementation from receiving and maintaining parallel connections to a single ground system. This would not be operationally advisable, however, In the event that a service from a ground system (e.g. update) is requested when all of the available CM connections are used, the service request would not be able to be fulfilled. A lower layer protocol error would result, which will then be interpreted by the dialogue service provider and indicated to the user via a D-START confirmation.

2.5.1.7 *Ground CM Connections*

2.5.1.7.1 A ground system will probably need to support more than one CM connection. The numbers in [3] illustrate this probability. However, there may be implementation issues that will reduce the number of connections needed. For instance, having ground-ground forwarding capability would reduce the number of CM connections needed by ground systems. Since other interested ground systems would receive the forwarded aircraft's application information, there may be no need to make CM connections to the aircraft.

2.5.1.7.2 There are also considerations that can drive the number of CM connections higher. In regions where there is an operational requirement to maintain a CM dialogue with each aircraft, then that ground system will need to allow for one CM connection per expected aircraft. This may lead to a number of simultaneous connections that will need to be maintained by the ground system. If the expected number of aircraft per ATSU are as in [3], this can lead to significant cost and performance concerns.

2.5.1.7.3 There are some factors that should be taken into account when designing a ground implementation. The operational need for maintaining a dialogue should be investigated, as well as traffic loading for the area in question. If there is not an operational need for maintaining a dialogue, costs may be minimized by not supporting this feature. There may also be tradeoffs that can be made for the number of connections, such as allowing a time interval for logons/updates to be completed. This could allow sequential, one-shot dialogues instead of parallel maintained dialogues.

2.5.1.7.4 A ground CM implementation should take into account the possibility of receiving multiple connection requests at the same time. There is nothing in the CM SARPs that prohibit a CM implementation from receiving and maintaining parallel connections to a single aircraft. This would not be operationally advisable, however, In the event that a service from an aircraft (e.g. logon) is requested when all of the available CM connections are used, the service request would not be able to be fulfilled. A lower layer protocol error would result, which will then be interpreted by the dialogue service provider and indicated to the user via a D-START confirmation.

2.6 Indexes/Tables

2.6.1 ASN.1 Type Index

2.6.1.1 Table 2.6-1 lists each ASN.1 type defined in the CM SARPs in alphabetical order.

2.6.1.2 The second column lists those ASN.1 types that are used in the definition of the ASN.1 type in the first column, and the third column lists those ASN.1 types that use it. The second and third columns are therefore inverse references.

2.6.1.3 The range and resolution at the primitive level are given where applicable in the fourth column.

Table 2.6-1. ASN.1 Type Index

Type	Types Used by this Type	Types Using this Type	Range and Resolution
AircraftFlightIdentification		CMLogonRequest	
Airport		CMLogonRequest	
AEQualifier		AEQualifierVersion AEQualifierVersionAddress	
AEQualifierVersion	AEQualifier VersionNumber	CMLogonRequest CMLogonResponse	
AEQualifierVersionAddress	AEQualifier VersionNumber APAddress	CMLogonRequest CMLogonResponse	
APAddress	LongTsap ShortTsap	AEQualifierVersionAddress	
CMAbortReason		CMAircraftMessage CMGroundMessage	
CMAircraftMessage	CMLogonRequest CMContactResponse CMAbortReason		
CMContactRequest	FacilityDesignation LongTsap	CMGroundMessage	
CMContactResponse	Response	CMAircraftMessage	
CMForwardRequest	CMLogonRequest	CMGroundMessage	
CMForwardResponse		CMGroundMessage	
CMGroundMessage	CMLogonResponse CMContactRequest CMForwardRequest CMForwardResponse CMUpdate CMAbortReason		
CMLogonRequest	AircraftFlightIdentification LongTsap AEQualifierVersionAddress AEQualifierVersion FacilityDesignation	CMAircraftMessage CMForwardRequest	

Type	Types Used by this Type	Types Using this Type	Range and Resolution
	Airport DateTime		
CMLogonResponse	AEQualifierVersionAddress AEQualifierVersion	CMGroundMessage CMForwardResponse CMUpdate	
CMUpdate	CMLogonResponse	CMGroundMessage	
Date	Year Month Day	DateTime	
DateTime	Date Time	CMLogonRequest	
Day		Date	Range: 1 - 31 Resolution: 1
FacilityDesignation		CMContactRequest CMLogonRequest	
LongTsap	ShortTsap	APAddress CMContactRequest CMLogonRequest	
Month		Date	Range: 1 - 12 Resolution: 1
Response		CMContactResponse	
ShortTsap		APAddress LongTsap	
Time	Timehours Timeminutes	DateTime	
Timehours		Time	Range: 0 - 23 Resolution: 1
Timeminutes		Time	Range: 0 - 59 Resolution: 1
VersionNumber		AEQualifierVersion AEQualifierVersionAddress	
Year		Date	Range: 1996 - 2095 Resolution: 1
… (extensibility marker)			CMAircraftMessage CMGroundMessage CMAbortReason

2.6.2 Message Content Glossary

2.6.2.1 The following messages and message elements as used in the CM SARPs, shown here in alphabetical order:

AEQualifier An integer from 0-255 that identifies a particular application.

AEQualifierVersion An indication of 1-256 airborne data link applications. Consists of AEQualifier and VersionNumber.

AEQualifierVersionAddress An indication of 1-256 airborne data link applications. Consists of AEQualifier, VersionNumber, and APAddress.

Aircraft Flight Identification Field 7 of the ICAO flight plan.

Airport	An IA5 string of 4 characters indicating an airport’s identification.
APAddress	An application’s unique technical communications address, which is either a long or short TSAP.
CMAbortReason	Specifies the reason of an abort caused by the CM ASE.
CMAircraftMessage	Contains the identification and the actual contents of an aircraft-generated CM message.
CMContactRequest	Identifies the facility that the aircraft is to perform a CM logon with.
CMContactResponse	Contains the result of the attempted contact.
CMForwardRequest	Contains the aircraft logon information that is to be ground-forwarded to a specific facility.
CMForwardResponse	Contains the result of the attempted forward.
CMGroundMessage	Contains the identification and the actual contents of a ground-generated CM message.
CMLogonRequest	Contains the appropriate aircraft addressing, application and flight data necessary to perform a CM logon.
CMLogonResponse	Contains the appropriate ground application information in response to a logon request.
CMUpdate	Contains the appropriate ground application information that will replace the application information currently held by the aircraft.
Date	Provides the date expressed in year, month and day.
DateTime	Provides the date and time.
Day	Specifies the day of the month.
FacilityDesignation	Provides the 4 to 8 character ICAO facility designation.
LongTsap	Provides the application address consisting of the RDP and Short TSAP.
Month	Specifies the month of the year
Response	Specifies whether a contact was successful or unsuccessful.
ShortTsap	Provides the application address consisting of the ARS, LOC, SYS, NSEL and TSEL.
Time	Provides the time expressed in hour and minutes.
Timehours	Specifies time in hours of a day.
Timeminutes	Specifies time in minutes of an hour.
VersionNumber	Specifies the version number of an application.
Year	Specifies the year between 1996 and 2095.

2.7 Example Scenarios

2.7.1 Introduction

2.7.1.1 This section contains a set of example scenarios that illustrate how CM might be used. The purpose of this section is to demonstrate some scenarios that are theoretically possible using CM. It is not meant to indicate what is required from an operational point of view.

2.7.2 CM Logon

2.7.2.1 An aircraft's avionics initiates a CM-logon request. The aircraft carries CPDLC and ADS applications, and is capable of both receiving and sending a CPDLC-start. Therefore, it provides the aircraft's CPDLC address as well as the ADS address, along with the application identifier (AEQualifier) and version number for each. Based on the routing domain particulars, long TSAPs are used to specify the addresses of both applications. In order to ensure that the aircraft can be properly converted, the departure and destination airports are also given.

2.7.2.2 The ground system receives the aircraft's logon data. The ground system wants to control the use of CPDLC dialogues by not allowing aircraft to initiate a CPDLC-start. Therefore the ground system only returns the CPDLC application identifier (AEQualifier) and version number to the aircraft for CPDLC, along with the ADS application identifier and version number. The ground system uses the provided information to correlate the aircraft with a flight plan held in its flight data base. The ground system then distributes the aircraft's ADS and CPDLC information to the appropriate software processes within the system, where the processes determine that the versions of the ADS and CPDLC carried by the aircraft are compatible. The ground system also retains the aircraft's logon information for future ground forwarding purposes. The appropriate operators' displays will be updated to note that the aircraft has successfully completed a CM logon.

2.7.2.3 Upon receipt of the logon response from the ground system. The avionics distributes the information to the appropriate software processes. The version numbers are checked, and it is determined that the aircraft's CPDLC and ADS versions are compatible with the ground system's. Since no CPLDC address was given, the pilot's displays will be updated to reflect that the pilot may not initiate a CPDLC dialogue.

2.7.3 CM Forward and CM Update

2.7.3.1 A ground system that support ground forwarding determines from an aircraft's flight plan that a neighbouring centre will require to perform data link services with that aircraft. Therefore, the Flight Data Processor retrieves the aircraft's logon information and ground forwards it to the centre.

2.7.3.2 The receiving centre, which supports the use of ground forwarding information, receives the forwarded data and correlates the information with the proper flight plan held in its flight plan data base. The aircraft's address information is distributed to the various software processes as appropriate. The ground system then performs a CM update with the aircraft, and will provide the ground system's application information that corresponds to that received in the forwarded information and the specific dictated by the flight plan and needed service.

2.7.3.3 Upon receipt of the CM update information, the avionics will distribute the application information to the appropriate software processes where version number checking is performed. The pilot's display will be updated to inform the aircrew that information for a new ground system has been received, and that the data link services (if any) are now available.

2.8 Example Encoding

2.8.1 CMLogonRequest PDU

2.8.1.1 The following is an example of the encoding of a CMLogonRequest PDU. It contains an address with a long TSAP for ADS and a short TSAP for CPDLC.

Element	Sub-element	Value	Encoding	Comments
CMAircraftMessage CHOICE		CMLogonRequest [0]	0 00	no extension choice 0
SEQUENCE			100000	option [2] enable
aircraftFlightIdentification [0]		UAL123	100 0101 0101 0100 0001 0100 1100 0011 0001 0011 0010 0011 0011	6 characters U A L 1 2 3
CMLongTSAP [1]				
SEQUENCE				
rDP		c155414c00	1100 0001 0101 0101 0100 0001 0100 1100 0000 0000	c1 55 41 4c 00
shortTSAP				
SEQUENCE			1	option [0] enable
ars [0]		000001	0000 0000 0000 0000 0000 0001	1
locSysNselTsel [1]		00110202a5fabcde00 0111	1 0000 0000 0001 0001 0000 0010 0000 0010 1010 0101 1111 1010 1011 1100 1101 1110 0000 0000 0000 0001 0001 0001	eleven octets 00 11 02 02 a5 fa bc de 00 01 11
groundInitiated [2]				
SEQUENCE SIZE (2)			0000 0001	size 2
AEQualifier		0	0000 0000	ADS
VersionNumber		1	0000 0000	
APAddress				
CHOICE [1]			1	choice [1]

Element	Sub-element	Value	Encoding	Comments
shortTSAP				
SEQUENCE			1	option [0] enable
ars [0]		000001	0000 0000 0000 0000 0000 0001	1
locSysNselTsel [1]		0011025fa5fabcde00 0111	1 0000 0000 0001 0001 0000 0010 0101 1111 1010 0101 1111 1010 1011 1100 1101 1110 0000 0000 0000 0001 0001 0001	eleven octets 00 11 02 5f a5 fa bc de 00 01 11
AEQualifier		2	0000 0010	CPDLC
VersionNumber		1	0000 0000	
APAddress				
CHOICE [1]			1	short TSAP
shortTSAP				
SEQUENCE			1	option [0] enable
ars [0]		000001	0000 0000 0000 0000 0000 0001	1
locSysNselTsel[1]		0011035fa5fabcde00 0111	1 0000 0000 0001 0001 0000 0011 0101 1111 1010 0101 1111 1010 1011 1100 1101 1110 0000 0000 0000 0001 0001 0001 xxxx	eleven octets 00 11 03 5f a5 fa bc de 00 01 11

2.8.1.2 Thus the PDU is encoded as 61 hexadecimal octets:

```
10 45 54 14 c3 13 23 3c 15 54 14 c0 08 00 00 0c 00 44 08 0a 97 ea f3 78 00 04 44 04 00 03 00 00 01 80 08
81 2f d2 fd 5e 6f 00 00 88 81 00 60 00 00 30 01 10 35 fa 5f ab cd e0 00 11 10
```

2.8.2 CMLogonResponse PDU

2.8.2.1 The following is an example of the encoding of a CMLogonResponse PDU. It contains an address with a long TSAP for CPDLC and the Qualifier and Version Number for ADS.

Element	Sub-element	Value	Encoding	Comments
CMGroundMessage CHOICE		CMLogonResponse [0]	0 00	no extension choice 0
SEQUENCE			11	option [0] and [1] enable
AirInitiatedApplications[0]				
SEQUENCE SIZE(1)			0000 0000	size 1
AEQualifier		2	0000 0010	CPDLC
VersionNumber		1	0000 0000	
APAddress				
CHOICE [0]			0	long TSAP, choice [0]
longTSAP				
SEQUENCE				
rDP		c144455500	1100 0001 0100 0100 0100 0100 0101 0101 0000 0000	c1 44 45 55 00
shortTSAP				
SEQUENCE			1	option [0] enable
ars [0]		000101	0000 0000 0000 0001 0000 0001	101
locSysNselTsel [1]		00110c12a5fabcde000111	1 0000 0000 0001 0001 0000 1100 0001 0010 1010 0101 1111 1010 1011 1100 1101 1110 0000 0000 0000 0001 0001 0001	eleven octets 00 11 0c 12 a5 fa bc de 00 01 11
groundOnlyInitiated [1]				
SEQUENCE SIZE (1)			0000 0000	size 1
AEQualifier		0	0000 0000	ADS
VersionNumber		1	0000 0000 xxxx xxx	

2.8.2.2 Thus the PDU is encoded as 27 hexadecimal octets:

18 00 08 01 82 88 8a aa 01 00 01 01 80 08 86 09 52 fd 5e 6f 00 00 88 80 00 00 00

2.8.3 CMContactRequest PDU

2.8.3.1 The following is an example of the encoding of a CMContactRequest PDU. It contains the facility designation and long TSAP of the facility that the aircraft is to logon to.

Element	Sub-element	Value	Encoding	Comments
CMGroundMessage CHOICE		CMContactRequest [2]	0 10	no extension choice 2
SEQUENCE				
facilityDesignation		CYVRAPP1	100 0100 0011 0101 1001 0101 0110 0101 0010 0100 0001 0101 0000 0101 0000 0011 0001	8 characters C Y V R A P P 1
longTSAP				
SEQUENCE				
rDP		c143414e00	1100 0001 0100 0011 0100 0001 0100 1110 0000 0000	c1 43 41 4e 00
shortTSAP				
SEQUENCE			1	option [0] enable
ars [0]		020101	0000 0010 0000 0001 0000 0001	20101
locSysNselTsel [1]		00110a23a5fabcde000111	1 0000 0000 0001 0001 0000 1010 0010 0011 1010 0101 1111 1010 1011 1100 1101 1110 0000 0000 0000 0001 0001 0001 xxxx xxx	eleven octets 00 11 0a 23 a5 fa bc de 00 01 11

2.8.3.2 Thus the PDU is encoded as 27 hexadecimal octets:

28 86 b2 ac a4 82 a0 a0 63 82 86 82 9c 01 02 01 01 80 08 85 11 d2 fd 5e 6f 00 00 88 80

3. ADS APPLICATION

3.1 Introduction

3.1.1 Purpose

3.1.1.1 In line with normal ICAO practice, this document was developed as a companion document to the ATN Automatic Dependent Surveillance — ADS — SARPs. It may be read alongside the ATN ADS SARPs, in order to provide a greater understanding of the specification itself. Alternatively, readers who simply want to understand the purpose of the ADS Application rather than the detail of the specification may read it instead of the ATN ADS SARPs.

3.1.1.2 This document also provides some historical information on the development of the ADS Application and explanations as to why the ADS Application is specified the way it is, including corresponding notes and recommendations, in the SARPs.

3.1.2 Scope

3.1.2.1 This document provides guidance material for those implementing of the Automatic Dependant Surveillance as part of the ATN Flight Information Service Application.

3.1.2.2 This document does not define any mandatory or optional requirements for the ADS Application, neither does it define any recommended practices. This document does not instruct users on how to use the ADS Application in a particular operational environment.

3.1.2.3 The ADS SARPs are dedicated to Air Traffic Services. Aeronautical Operational Control (AOC) may choose to use the ADS SARPs as a model for their own applications.

3.1.3 History

3.1.3.1 The ADS Panel has forged the requirements for the ADS application over a period of several years. These are laid out in the *Manual of Technical Provisions for the ATN* (Doc 9705) [4]. The ADS SARPs were based on the operational requirements specified in this manual.

3.1.3.2 The application SARPs were validated through a number of methods, but chiefly through the development of independent implementations of the applications, and the successful running of interoperability trials between them.

3.1.4 Structure of Guidance Material

3.1.4.1 Chapter 3.1 — INTRODUCTION — contains the reason for providing guidance material as well as the scope. In addition, it provides a brief overview of ADS functionality, ADS's relationship with other SARPs, and identifies applicable reference documents.

3.1.4.2 Chapters 3.2 — OVERALL GENERAL FUNCTIONALITY — describes generic concepts that are used throughout the ADS SARPs and guidance material. This chapter also covers some implementation issues that are not addressed in the SARPs.

3.1.4.3 Chapter 3.3 — ADS SERVICE DESCRIPTION — gives a functional breakdown of the various services that ADS provides. It describes a peer to peer interaction, including reasons for why particular information is used or not used, and what actions on the information are expected.

3.1.4.4 Chapter 3.4 — ADS SARPs Section Description — clarifies any functionality that was not addressed in Chapter 3 on a chapter by chapter basis.

3.1.4.5 Chapter 3.5 — DIMENSIONS — gives some sample encoding sizes for guidance on what capacities need to be allowed for in order to meet normal operational expectations.

3.1.4.6 Chapter 3.6 — INDEXES / TABLES — contains an index of the ASN.1.

3.1.4.7 Chapter 3.7 — EXAMPLE SCENARIOS — gives some examples that describe typical scenarios one can expect in course of normal ADS operation.

3.1.4.8 Chapter 3.8 — EXAMPLE ENCODING — outlines some actual sample PER encoding of typical ADS messages.

3.1.5 ADS Application Overview

3.1.5.1 *Summary*

3.1.5.1.1 The ADS application is designed to give automatic reports to a user, that are derived from on-board navigation and position-fixing system, including aircraft identification, four-dimensional position and additional data as appropriate. The ADS reports give positional as well as other information likely to be of use to the air traffic management function, including air traffic control. The aircraft provides the information to the user under one of four circumstances:

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

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

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

d) under emergency conditions the aircraft provides information on a regular basis with no prior agreement with the ground system (known as an emergency contract); an event or periodic contract must already exist before an emergency contract can be established.

3.1.5.1.2 In addition, the ADS application provides a means to forward ADS reports from the ground system that has contracts with an aircraft, to another ground system.

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

3.1.5.2 *Establishment and Operation of a Demand Contract*

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

3.1.5.2.2 Any number of demand contracts may be sequentially established with an aircraft. Basic information is sent with the report (see 1.5.2.5). Optionally, at the request of the ground system, other information may also be sent (see 1.5.2.6).

3.1.5.2.3 The ground system sends a demand contract request to the avionics. This contains an indication of which optional information is required. The avionics then determines whether or not it is able to comply with the request. If the avionics can comply with the demand contract request, it sends the report as soon as possible. If the avionics cannot comply with the request, it sends a negative acknowledgement to the ground system indicating the reason for its inability to accept the contract.

3.1.5.2.4 If the avionics can partially comply with the request, it sends a noncompliance notification accepting those parts of the contract with which it can comply.

3.1.5.2.5 Each report always contains the following basic information:

a) the 3-D position of the aircraft;

b) the time;

c) an indication of the aircraft's navigational accuracy and ACAS status (figure of merit).

3.1.5.2.6 The demand contract stipulates which of the optional information fields are to be included in the ADS report. Optionally, a report contains an indication of:

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

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

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

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

e) the aircraft address (this is the 24 bit address unique to each airframe — not a network address);

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

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

3.1.5.3 *Establishment and Operation of an Event Contract*

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

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

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

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

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

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

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

e) Airspeed change. This is triggered when the aircraft's airspeed differs negatively or positively from its value at the time of the previous ADS report containing an air vector, by an amount which is equal to the airspeed change threshold which is specified in the event contract request.

f) Ground speed change. This is triggered when the aircraft's ground speed differs negatively or positively from its value at the time of the previous ADS report containing a ground vector, by an amount which is equal to the ground speed threshold which is specified in the event contract request.

g) Heading change. This is triggered when the aircraft's heading differs negatively or positively from its value at the time of the previous ADS report containing an air vector, by an amount which is equal to the heading change threshold which is specified in the event contract request.

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

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

j) Track angle change. This is triggered when the aircraft's track angle differs negatively or positively from its value at the time of the previous ADS report containing a ground vector, by an amount which is equal to the track angle change threshold which is specified in the event contract request.

k) Level change. This is triggered when the aircraft's level differs negatively or positively from its value at the time of the previous ADS report, by an amount which is equal to the level change threshold which is specified in the event contract request.

3.1.5.3.4 Acceptance of an event contract request implicitly cancels an existing event contract, if one exists. That is, there is no concept of modifying an existing event contract.

3.1.5.3.5 The ground system sends an event contract request to the avionics. This contains the type(s) of event(s) to be reported on, and the necessary parameters for that event (e.g. if the event is an level range deviation, then the upper and lower thresholds must be sent). The avionics then determines whether or not it is able to comply with the request. If the avionics can comply with the event contract request it sends a positive acknowledgement immediately (possibly as part of an ADS report). If the contracted event occurs, an ADS report is sent.

3.1.5.3.6 If the avionics cannot comply with the request, it sends a negative acknowledgement to the ground system indicating the reason for its inability to accept the contract.

3.1.5.3.7 If the avionics can partially comply with the request, it sends a noncompliance notification accepting those parts of the contract with which it can comply.

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

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

a) lateral deviation change — containing the lateral deviation threshold;

b) vertical rate change — containing the vertical rate threshold;

c) leaving a given level range — containing the upper and lower level thresholds;

d) way-point change — containing no further clarifying information;

e) air speed change — containing the airspeed change threshold;

f) ground speed change — containing ground speed change threshold;

g) heading change — containing heading change threshold;

h) extended projected profile change — containing either a projected time or a number of way points;

i) figure of merit change — containing no further clarifying information;

j) track angle change — containing the track angle change threshold;

k) level change — containing level change range.

3.1.5.3.10 The choice of additional optional information blocks in the ADS report is made as follows:

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

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

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

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

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

3.1.5.3.11 Some event contracts require the air system to report when some parameter differs from a previously reported value by some amount. If one or more of these event types are requested in the contract (and the air system can support them), then the air system will send back a baseline report at the time it accepts the contract. This baseline report contains values against which the event will be measured. For example, if the heading-change event is used, with a value of four degrees in the event contract, then the air system will immediately return a baseline report containing the air vector and ground vector. At some time in the future, if the aircraft's heading varies from the value stated in the air vector by more than four degrees, an event report will be triggered. Events that require a baseline report to be sent are:

a) air-speed-change;

b) ground-speed-change;

c) heading-change;

d) track-angle-change;

e) level-change.

3.1.5.4 *Establishment and Operation of a Periodic Contract*

3.1.5.4.1 This function allows the ground system to establish a periodic contract with the aircraft, and then for the conditions of that contract to be realised. Realisation of the contract involves the sending of reports from the aircraft to the ground system at regular intervals (the reporting rate). Only one periodic contract may exist between a ground system and the avionics at any one time.

3.1.5.4.2 A set of basic information is sent with every report. Optionally, at the request of the ground system, other information may also be sent; furthermore they may be sent at a time interval which is a multiple of the reporting rate. The contract that is agreed includes the reporting rate, the optional information to be sent and the rate at which they are to be sent.

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

3.1.5.4.4 Acceptance of a periodic contract request implicitly cancels an existing periodic contract, if one exists. That is, there is no concept of modifying an existing periodic contract.

3.1.5.4.5 If the avionics cannot accept the contract, it sends a negative acknowledgement to the ground system indicating the reason for its inability to accept the contract.

3.1.5.4.6 If the avionics can partially comply with the request, it sends a noncompliance notification accepting those parts of the contract with which it can comply.

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

a) reporting interval;

b) flight id modulus;

c) projected profile modulus;

d) ground vector modulus;

e) air vector modulus;

f) weather modulus;

g) airframe id modulus;

h) navigational intent modulus and projection time;

i) extended projected profile modulus and projection time or the amount of information required.

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

3.1.5.5 *Cancellation of Contracts*

3.1.5.5.1 This function allows the ground system explicitly to cancel a contract that is in operation. The ground system sends a cancel contract message to the avionics. The avionics cancels the contract and acknowledges the cancellation.

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

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

3.1.5.6 *Establishment and Operation of Emergency Contracts*

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

3.1.5.6.2 Any existing periodic contract is suspended pending the cancellation of the emergency contract. Normal operation of event contracts is maintained. Initially the emergency reporting rate is the lesser of 60 seconds or half any existing periodic contract rate (if one exists). The position, time and FOM are sent with each ADS report, and the aircraft address and ground vector sent with every fifth.

3.1.5.6.3 The avionics send reports to all ground systems with which it has event or periodic contracts at the emergency reporting rate.

3.1.5.6.4 The ADS emergency report has the same structure as normal ADS reports, except that the optional elements are fixed as described above.

3.1.5.7 *Modifying an Emergency Contract*

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

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

3.1.5.7.3 If the avionics is unable to change the reporting rate, the avionics will send a negative acknowledgement.

3.1.5.8 *Cancellation of Emergency Contract*

3.1.5.8.1 This function allows the aircraft to cancel the emergency contracts.

3.1.5.8.2 The avionics sends a cancel emergency message to the ground. If there was a periodic contract in place before the emergency was declared, then it is reinstated immediately after the emergency mode is cancelled. When the aircraft performs this function, emergency contracts with each ground system that it is in contact with are cancelled.

3.1.5.8.3 It is possible that, during the operation of an emergency contract, the ground system either replaces the periodic contract (using ADS-periodic-contract request), or cancels a periodic contract (using ADS-cancel). It may also be possible that no periodic contract is in place and, during the operation of an emergency contract, the ground system requests a new periodic contract (using ADS-periodic-contract request). It may also be possible that several of these events occur sequentially during the operation of an emergency contract.

3.1.5.8.4 In any of these cases, once the emergency contract is cancelled by the aircraft system, the latest of these requests is put into operation. If the latest thing to happen is a new ADS-periodic-contract request, then it will be brought into affect immediately. If the latest thing to happen is an ADS-cancel request (for the periodic contract), then no periodic contract will be re-instated on completion of the emergency contract.

3.1.5.9 *ADS Report Forwarding*

3.1.5.9.1 This function provides a method for a ground system to forward ADS reports received from an aircraft to another ground system. This function is initiated by a ground system having established one or more successful ADS contracts.

3.1.5.9.2 The initiating system sends an ADS start forward request to the receiving system; this may contain the first ADS report.

3.1.5.9.3 Having established communication, the initiating system can forward as many reports as it needs to. An ADS forward report contains the identification of the aircraft that the report is related to and either a periodic, event, demand, or emergency report.

3.1.5.9.4 When it wishes to break the communication link, the initiating system sends an ADS end forward request. If, at any time during the communication, the receiving system wishes to break the communication link, it must abort the connection.

3.1.5.10 *Aborts*

3.1.5.10.1 This function allows the airborne system, the ground system or the communications system to abort a connection in cases where a serious problem has occurred.

3.1.5.10.2 If the communications part of the airborne or ground system, or the network itself, detects an error, either in itself or in the protocol arriving from its peer, it will initiate an ADS provider abort.

3.1.5.10.3 If the user part of the airborne or the ground system detects an error, it has the option of initiating an ADS user abort.

3.1.5.10.4 In either of these cases, the result is that the connection is closed down immediately. Some of the messages already transmitted, but not yet confirmed, may be lost in transit. There is nothing to prevent the ground system attempting to make contact with the aircraft immediately following this.

3.1.6 Inter-Relationships With Other SARPs

3.1.6.1 There is no interaction between the ADS SARPs and the other CNS/ATM-1 Applications SARPs.

3.1.6.2 The ADS SARPs make use of the Upper Layer Communications Service, Annex 10, Volume III, Part 1, Chapter 3 (ATN) and Sub-volume IV of the *Manual of Technical Provisions for the ATN* (Doc 9705) [3] to perform dialogue service functions required by the ADS Application.

3.1.7 Structure of SARPs

3.1.7.1 All the air-ground SARPs are produced to a standard format. This has greatly helped the maintenance of document stability, commonality and presentation. The ADS SARPs are no different in basic layout from all other air-ground applications SARPs, except that the air-ground aspects of the application are in a separate section (2.2.1) to the ground-ground parts (2.2.2).

3.1.7.2 The ADS SARPs constitute the second part of sub-volume 2.

3.1.7.3 SARPs Section 2.2.1.1 — INTRODUCTION — gives a very brief, high level description of ADS, as an application enabling ADS services to be provided to a pilot via the exchange of messages between aircraft avionics and ground ADS systems. Since this overview contains no information directly related to the stipulation of specific standards, it is almost entirely written as series of informative notes.

3.1.7.4 SARPs Section 2.2.1.2 — GENERAL REQUIREMENTS — contains information and high level requirements for error processing and ADS version number requirements.

3.1.7.5 SARPs Section 2.2.1.3 — ABSTRACT SERVICE — defines the abstract service interface for the ADS Application. The ADS Application Service Element (ADS-ASE) abstract service is described from the viewpoint of the ADS-air-user, the ADS-ground-user and the ADS-service-provider.

3.1.7.6 SARPs Section 2.2.1.4 — FORMAL DEFINITION OF MESSAGES — describes the contents of all permissible ADS messages through definition of the ADS ASN.1 abstract syntax. All possible combinations of message parameters and their range of values are detailed.

3.1.7.7 SARPs Section 2.2.1.5 — PROTOCOL DEFINITION — splits up the specification of the ADS protocol into three parts: sequence diagrams for the services covered by the abstract service, protocol descriptions and error handling for the ADS Air and Ground-ASEs, and State Tables.

3.1.7.8 SARPs Section 2.2.1.6 — COMMUNICATION REQUIREMENTS — specifies the use of Packed Encoding Rules (PER) to encode/decode the ASN.1 message structures and stipulates the Dialogue Service requirements, including Quality of Service (QoS).

3.1.7.9 SARPs Section 2.2.1.7 — ADS USER REQUIREMENTS — describes the requirements imposed on the ADS-users concerning ADS messages and interfacing with the ADS-ASEs.

3.1.7.10 SARPs Section 2.2.1.8 — SUBSETTING RULES — specifies conformance requirements, which all implementations of the ADS protocol obey. The protocol options are tabulated, and indication is given as to whether mandatory, optional or conditional support is required to ensure conformance to the SARPs. These subsetting rules will permit applications to be tailored to suit individual ground implementations, commensurate with the underlying task, while still maintaining an acceptable level of interoperability.

3.1.7.11 SARPs Section 2.2.2 — ADS REPORT FORWARDING APPLICATION — is structured in eight chapters in the same way as 2.2.1.

3.1.8 References

[1] Automatic Dependent Surveillance Application, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705), Section 2.2.1

[2] Automatic Dependent Surveillance Report Forwarding Application, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705), Section 2.2.2

[3] Upper Layer Communications Service, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume IV of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[4] *Manual of Technical Provisions for the ATN* (Doc 9705)

[5] Introduction and System Level Requirements, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume I of the *Manual of Technical Provisions for the ATN* (Doc 9705)

3.2 Overall General Functionality

3.2.1 General

3.2.1.1 ADS is defined as two different applications — one handling the air-ground exchanges and the other handling the ground-ground exchanges. The former is called ADS, and the latter is called ADS Report Forwarding (sometimes abbreviated ARF).

3.2.2 Topology

3.2.2.1 *Air-Ground Link*

3.2.2.1.1 The primary connection that is made in ADS is between the aircraft and the ground system. The connection is always initiated by the ground system, by the process of setting up an ADS contract. In like manner, the ground system is always responsible for closing the connection down. During the connection, the airborne system is always reactive, apart from when an emergency is initiated. An emergency is initiated either by heuristics within the avionics, or by the pilot. In these circumstances, the airborne system automatically establishes an emergency contract with every ground system with which it has event or periodic contracts. For ground systems that have just requested a demand contract in a single shot, no emergency contract is established.

3.2.2.2 *Ground Forwarding*

3.2.2.2.1 A secondary connection type is available to allow forwarding of ADS reports on ground-based networks. One ATSU system may forward one or more ADS reports to another ATSU. The sending ATSU is responsible for both establishing and closing the connection. During the connection, the receiving system is purely a sink for information. No response is generated to anything. The only thing that the receiving system is capable of initiating is an abort.

3.2.2.3 *Overall Topology*

3.2.2.3.1 Figure 3.2-1 shows the overall topology of the ADS and ADS report forwarding applications. Each aircraft may have ADS contracts with a number of ground systems. Each ground system may have ADS contracts with a number of aircraft. When a ground system has one or more ADS contracts with an aircraft, these are supported over a transport connection between the two. Ground systems may also have connections with other ground systems in order to support the ADS report forwarding application.

3.2.2.3.2 Each air-ground connection can support several ADS contracts of different types at the same time. Thus if a ground system has an event and periodic contract with an aircraft, and also sends up occasional demand contracts, then all these will use the same connection.

3.2.2.3.3 Typically, a system will have some maximum limit of the number of connections it can have. An aircraft must be able to support at least four connections, with four different

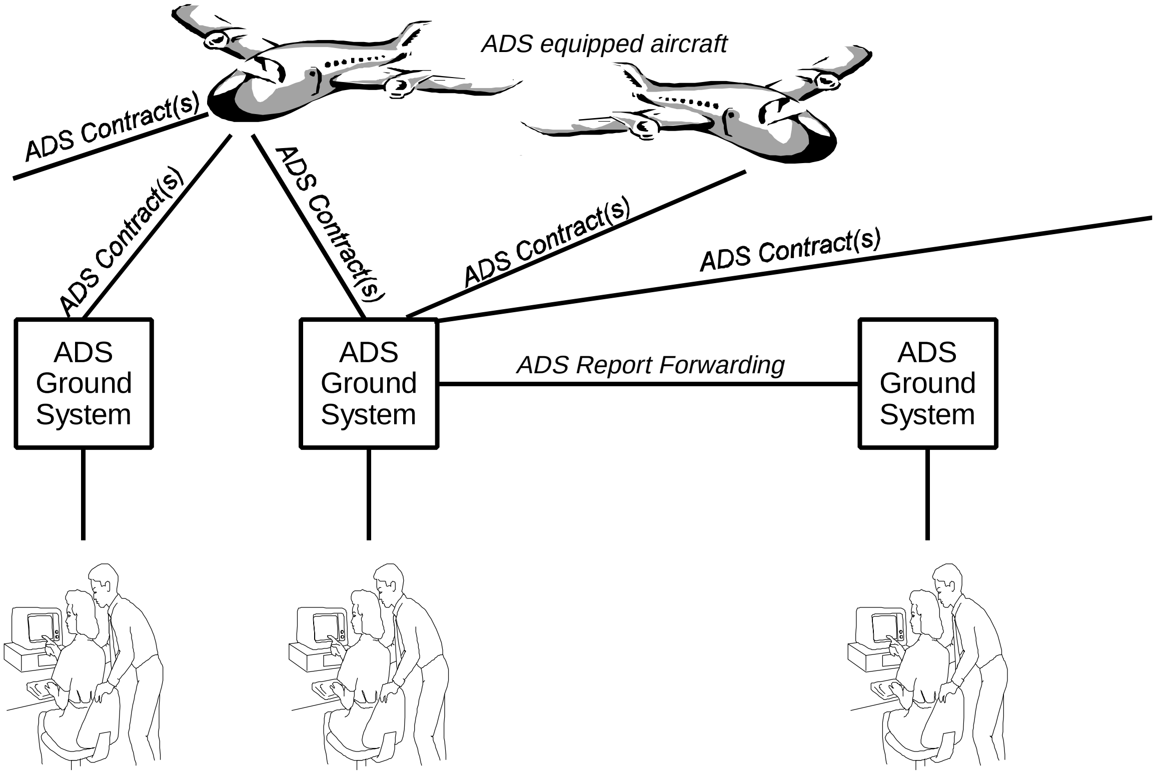


Figure 3.2-1. Overall Topology of the ADS and ADS Report Forwarding Applications

ground systems. However, there is no requirement that limits them to four. Operational scenarios have been identified that may require the use of all four connections for ATC purposes. It may be useful, for example, for an aircraft to support a fifth connection for AOC purposes — but at least four must be available for ATC purposes.

3.2.2.3.4 A ground system, on the other hand, will probably have a much larger maximum. The ADS application for a busy air traffic control centre may have to support hundreds of connections. In an area where there are few aircraft, there may only be a requirement for a few tens of connections.

3.2.2.3.5 So the maximum number of connections for a ground system is highly dependent upon the area where it is installed. An aircraft, on the other hand, must be able to support at least four, and there is no operational need that is anticipated for any more.

3.2.2.3.6 There is a requirement that an aircraft must be able to have contracts with at least four ground systems. However, there is nothing to prevent an aircraft system being able to support more than four. To avoid undue complication of the following text, the case of an aircraft supporting four ground systems has been assumed. The reader is invited to make his/her own adjustments if they are dealing with an aircraft that supports more than four ground systems.

3.2.2.3.7 In order for the airborne system to support contracts with four ground systems, it must be able to support at least five connections. Suppose that four ground systems (A, B, C and D) have contracts with an aircraft. A fifth ground system (E) issues a periodic contract (say) to the same aircraft. The airborne system must have the resources to receive the fifth connection, and then to refuse it. In doing so, it must inform ground system E that it is working at its maximum capacity, and it must also inform E of the ICAO facility designators of the systems A, B, C and D. Thus, if the aircraft is moving into the sector covered by E, the controller at E has enough information to call A, B, C or D (by telephone perhaps) and ask that they discontinue their contracts with the aircraft.

3.2.2.4 *Central Server Architecture*

3.2.2.4.1 The ADS application was designed to allow the direct connection of a ground system to an aircraft, and the subsequent forwarding of ADS reports to other ground systems. This provides the opportunity to develop a central ADS server for a region.

3.2.2.4.2 The concept of a central ADS server for a region is that a single ground system (the central ADS server) acts for all the ground based systems in the region for supply of ADS data. The central server would establish contracts with all aircraft in the region, and then forward reports to appropriate ground systems. For example, all reports concerning aircraft in a particular sector would be forwarded to the system operating for that sector.

3.2.2.4.3 For every aircraft entering the region, the central server would establish ADS contracts. Depending on the policy for the region, periodic and/or event reports could be established to monitor the progress of the aircraft. The central server would then be required to make a decision, for each ADS report received, where to forward it to. For example, it could be forwarded to the ground system where the aircraft's controller was operating, and, if the aircraft was close to a boundary, the neighbouring controller's systems as well.

3.2.2.4.4 The concept of a central server is beneficial since it reduces the use of air-ground bandwidth (only one connection is ever in place), and it only requires one system that has the full ADS functionality. All the other systems only require facilities to receive forwarded ADS reports.

3.2.2.4.5 In order for such a concept to work, the main constraint is that there must be a common policy regarding what contracts to set up. All systems must require the same type of contracts.

3.2.3 Abstract Internal Architecture

3.2.3.1 *Type of ASE in ADS*

3.2.3.1.1 The ADS application defines two types of ASE — the ADS-ASE and the ARF-ASE. The ADS-ASE has two variations — the ADS-air-ASE and the ADS-ground-ASE. These are used in all air to ground links. The ADS-ARF-ASE also has two variations — the initiating and the receiving ADS-ARF-ASE. These are used for the ADS report forwarding function.

3.2.3.2 *ADS-ASE Functionality*

3.2.3.2.1 The ADS-ASEs are responsible for all aspects of air-ground communication related to ADS. They encode and decode the PDUs. They maintain a state machine that only allows PDUs to be sent at an appropriate time, and detect if the peer application sends PDUs at an inappropriate time.

3.2.3.3 *ARF-ASE Functionality*

3.2.3.3.1 The ARF-ASEs are responsible for all aspects of ground based communication related to ADS. They encode and decode the PDUs. They maintain a state machine that only allows PDUs to be sent at an appropriate time, and detect if the peer application sends PDUs at an inappropriate time.

3.2.3.4 *ADS-User Functionality*

3.2.3.4.1 In the model of the ADS, there is a module called the ADS-user. The functionality of the ADS-user is described in 2.2.1.7. The ADS-ground-user is responsible for initiating the contracts and making the ADS-reports available to the end user. There are very few requirements placed on the ADS-ground-user. This allows the implementors a great deal of freedom in the method of implementation of this component.

3.2.3.4.2 The ADS-air-user is responsible for the operation of the ADS contracts. On receipt of the contract, it is responsible for responding to the request and then creating and submitting ADS reports in line with the contract. Most of the requirements for user functionality fall within the area of the ADS-air-user. There are few requirements for the ARF-users, and so the implementors have a great deal of scope in this area.

3.2.3.5 *Product Architecture*

3.2.3.5.1 The SARPs have defined an abstract model for the purposes of definition. That is, it has split the functionality between the ASE and the user and has defined an abstract interface between the two. It is strongly emphasised that there is no requirement on an implementor to build such an interface. If it is convenient, from an engineering perspective, to build an interface between two modules that embody the functionality of the ASE and the user, then the implementor is free to do so. However, if it is more convenient to build the system with interfaces in other places, then that is also acceptable. In testing a product to see if it conforms to the SARPs, no test can be made to test internal interfaces within the system.

3.2.3.5.2 The internal structure of the ATN ASE is not standardised across applications; for instance, ADS ASE is defined as several modules while CM is defined as a single module.

3.2.4 **Implementation Dependent Functionality**

3.2.4.1 The SARPs specify some of the requirements for the user, but leave a lot up to the implementors. There are no requirements that state how the user interface appears, how

ADS interacts with the flight data processing systems, how ADS interacts with higher level functionality and with other applications such as CM and CPDLC. All this is implementation dependent.

3.2.4.2 It must be expected that between the functionality specified in the SARPs and the human user, there will be a wealth of features, probably exceeding the functionality of the SARPs. For example, there may be sets of standard contracts that are used under certain conditions that are automatically set up without the knowledge of the controller. The controller's screen may show the positions of aircraft based on a mixture of radar and ADS; automated functions may use ADS to extract information from the aircraft's flight management system and compare it with information stored in the filed flight plan. None of these features are described in the SARPs.

3.2.5 Rationale for ASE/User Split

3.2.5.1 The rationale for the split in functionality between the ASE, the user and the implementation dependent parts is as follows:

3.2.5.2 The ASE contains all the functionality that is necessary to ensure the interoperability at the syntactic level. That is, two valid implementations of ASEs will be able to interact, passing data to each other in the correct order. They will be able to check the format of the data, ensure that it has been sent at an appropriate point in the dialogue and ensure that the peer ASE is behaving according to the requirements in the SARPs also. The ASE thus ensures interoperability.

3.2.5.3 The SARPs defines some requirements for the user. These are the minima that are required to ensure the semantic interoperability of the two peers. That is, it explains how the data that is transported by the ASE is interpreted. Thus it explains how each type of contract is interpreted and how it operates.

3.2.5.4 Some care has been taken to ensure that the requirements are not over-specific. That is, they do not specify things that are not absolutely essential to the syntactic and semantic interoperability of the ADS function. Different manufacturers can build this implementation dependent part in different ways, without effecting the interoperability between different implementations. This implementation dependent part has not been specified in the SARPs, but will be specified by individual product manufacturers or regional standards.

3.2.6 Inter-relationship with other ATN Applications

3.2.6.1 There is no interaction required between the ADS Application and the other ATN applications. The ADS application is a stand-alone application which can be developed, certified, installed and operated completely independently from the other ATN Applications.

3.2.6.2 There is a technical relationship between the ADS and CM applications. In order for a ground system to initiate the first contract with an aircraft, it must know the address of the ADS application in the aircraft. It is anticipated that this will normally be done using the CM application. When the aircraft approaches an FIR boundary, it will initiate CM. This will pass the address of the aircraft's ADS application to the ground system. Alternatively, the ground system from the previous FIR may pass the address of the ADS application using the CM-forward function. The ground system must have some way of storing the addresses it receives. Following this sequence of events, the ground system is able to initiate ADS contracts with the aircraft.

3.2.6.3 From a technical point of view, there is no relationship between the ADS application and either CPDLC or FIS. Operationally, they could be linked in a single service. For example, ADS could be used by the controller to obtain an extended projected profile. This could be used to compare the aircraft's flight plan with the filed flight plan. If there is a discrepancy, the ground system could used CPDLC to warn the aircrew of the discrepancy. Such operational linkage of the different applications is beyond the scope of the SARPs.

3.2.7 Ground ADS Exchanges

3.2.7.1 Functionality is defined in the SARPs to allow a ground system to forward the ADS reports that it has received to another ground system. The functionality of this ADS report forwarding function is very simple: The initiating system opens a connection with the receiving system. It then transmits ADS reports to the receiving system, and then closes the connection. Provided that the receiving system has implemented this functionality, it has no option but to receive the ADS reports. The only action the receiving system can take is to abort the connection.

3.2.7.2 The initiating system may send any ADS reports it has in any order. The reports are identified by the aircraft-id and each reports contain a time and date stamp. Thus the receiving system is able to identify the reports accurately, and is able to discard reports that are not relevant.

3.2.8 Dialogue Management

3.2.8.1 *General*

3.2.8.1.1 The term "dialogue" here refers to the end-to-end communication path provided by the Dialogue Service Provider. The Dialogue Service is described in detail in the ULA SARPs [3]. A dialogue is mapped directly onto a transport connection.

3.2.8.1.2 The dialogue is always initiated by the ground system, by the process of setting up an ADS contract. In like manner, the ground system is always responsible for closing the dialogue.

3.2.8.1.3 The dialogue service provider is used by the ADS-ASEs for the following purposes:

- establishment,
- graceful release and abort of a dialogue,
- transfer of data,

- support for quality of service and version number negotiation, and
- application naming.

3.2.8.2 ***Optimisation of the use of dialogues***

3.2.8.2.1 The ADS service hides the use of the underlying communication service to the ADS-users. The ADS users are aware when contracts are established and cancelled. However, they are not necessarily aware of when dialogues are started and ended.

3.2.8.2.2 Multiplexing over a single dialogue of ADS contracts set up between a ground and an airborne ADS system is supported by the ADS protocol. One demand contract, one event contract, one emergency contract and one periodic contract can be multiplexed in parallel on a single dialogue (although a periodic contract is suspended during the operation of an emergency contract). The time required to establish the dialogue before any operational data can be exchanged penalises only the first ADS contract. The dialogue multiplexing is performed by the Low Interface Module.

3.2.8.3 ***Dialogue Establishment***

3.2.8.3.1 When the establishment of a new ADS contract is requested by the ADS-ground-user, if no dialogue is already in place, then a dialogue is established. During this process, data related to the ADS contract are exchanged. During the time of the dialogue establishment, no new ADS service request can be accepted by the ADS-service provider (apart from ADS-user-abort).

3.2.8.3.2 On receipt of an ADS contract request, if a dialogue is already in place, it is used immediately for data transmission.

3.2.8.3.3 The SARPs prohibit an ADS-air-ASE from requesting the establishment of a dialogue.

3.2.8.4 ***Dialogue Release***

3.2.8.4.1 The dialogue is closed when there are no contracts left in place, that is, when all contract have been completed or cancelled. Without explicit action from the pilot or a ground operator, the ground ADS system triggers the release of the dialogue.

3.2.8.4.2 The dialogue can be abruptly terminated by an abort condition from the ADS-ground-ASE at any time after the D-START request has been issued by the ADS-ground-ASE, and by the ADS-air-ASE at any time after the D-START indication has been received.

3.2.8.4.3 The SARPs prohibit an ADS-air-ASE from requesting the release of a dialogue.

3.2.9 Protocol Monitoring

3.2.9.1 *General*

3.2.9.1.1 The ADS ASE controls that the protocol is correctly handled by the peer and can be correctly operated locally. The generation and transmission of expected responses are monitored by the peer ADS ASE.

3.2.9.1.2 In case a serious error is detected, the dialogue in place with the peer ADS system is aborted and all contracts in place or being established are cancelled. Active ADS-users are informed of this situation by a ADS-provider-abort indication and are provided with a reason for the abort. These cases are described in the following sections.

3.2.9.2 *Exception Handling*

3.2.9.2.1 **Service Timers**

3.2.9.2.1.1 In case of a confirmed service (ADS-demand-contract, ADS-event-contract, ADS-periodic-contract, ADS-cancel-contract, ADS-cancel-all-contracts, ADS-modify-emergency-contract, and ADS-cancel-emergency-contract), the generation and the transmission of the response expected to be issued by the remote ADS-user or ADS-ASE are monitored by the ADS ASEs by activating a timer. If the APDU corresponding to the confirmation or ADS-report indication is not received by the requesting ADS ASE within a locally specified period of time, the dialogue is aborted. Both ADS-users are informed of this situation by a ADS-provider-abort indication with a reason “timer expiry”. This timer is a technical service timer and is not directly connected to any operational timers set, except that it ought to be sufficiently larger than any operational timer to prevent “nuisance” timer-expiration aborts.

3.2.9.2.1.2 The reception of the D-END confirmation is also monitored. If not received, the ultimate action is for the ADS-ground ASE to try to abort the dialogue. As the ADS-users are not active any more, no ADS-abort indication is generated.

3.2.9.2.2 **An Unrecoverable Internal Error occurred**

3.2.9.2.2.1 The unrecoverable system error is intended to cover cases where a fault causes a system lockup or the system to become unstable (e.g. the system gets short of memory). If the system has enough resources to do it, it will abort the connection and inform both the local user and the peer of the situation with the abort reason “unrecoverable system error”.

3.2.9.2.2.2 The unrecoverable system error is written as a recommendation in the SARPs instead of a requirement, as it is recognised that, depending on the nature of the error in the system, it may not be possible to regain control in order to either perform an abort, or inform the user of the abort situation.

3.2.9.2.3 The received primitive or APDU was not expected.

3.2.9.2.3.1 On receipt of a dialogue service indication or confirmation, the ASE checks that the APDU transmitted in the user data parameter of this primitive is authorised for this particular dialogue service primitive. If the received APDU is not authorised, this means that the peer made an error and is not running the protocol correctly. The dialogue is therefore aborted by the ASE receiving the APDU with the reason “invalid PDU”.

3.2.9.2.3.2 On receipt of a dialogue service indication or confirmation, the ASE checks that the invoked dialogue service is authorised in its current state. If the primitive is not authorised, this means that the sequencing of the dialogue primitives defined for the ADS protocol has not been respected by the peer. The dialogue is therefore aborted by the ASE receiving the out of sequence primitive with the reason “sequence error”.

3.2.9.2.4 The Start dialogue has been rejected by the peer

3.2.9.2.4.1 When the ADS-air-ASE receives a dialogue start indication it is required to accept the dialogue. If the ADS-ground-ASE receives a rejection of the dialogue start which was initiated by the airborne system, the airborne ADS system has not operated correctly. An abort indication is given to the ADS-ground-user with reason “sequence error”.

3.2.9.2.4.2 If the ADS-ground-ASE receives a rejection of the dialogue start which was initiated by the dialogue service, an abort indication is given to the ADS-ground-user with reason “cannot establish contact”.

3.2.9.2.5 The End dialogue has been rejected by the peer

3.2.9.2.5.1 When all contracts are completed, the D-END request is invoked to shut down the dialogue. Upon receipt of the corresponding D-END indication, the ADS-air-ASE's only choice is to accept the D-END by setting the D-END response Result parameter to the abstract value “accepted”. If for some reason this value is not present (as checked in SARPs section 2.2.1.5.4.6.1), then the ADS-ground-ASE will abort the dialogue with reason “dialogue-end-not-accepted”. Since there are no active users any more, no indication is given to the ADS-ground-user.

3.2.9.2.6 The expected APDU is missing

3.2.9.2.6.1 With the exception of the D-END primitives, if the user data parameter of a received indication or confirmation does not contain an APDU, this means that the peer has not run the ADS protocol correctly. The ASE fails to decode an APDU, and so the dialogue is aborted by the ASE receiving the empty primitive with the reason “decoding error”.

3.2.9.2.7 The received APDU cannot be decoded

3.2.9.2.7.1 If the received PDU cannot be decoded, a decoding error is raised highlighting a transmission error or an encoding error by the APDU sender. The dialogue is therefore aborted by the ASE receiving the invalid APDU with the reason “decoding error”.

3.2.9.2.8 The requested QOS does not match the one specified in the FIS SARPs

3.2.9.2.8.1 The ATN Sub-Volume 1 (5) dictates the values used for application service priority and RER quality of service parameters for all ATN applications. These values must be adhered to so proper levels of flight safety and performance are maintained. For ADS, the values used are "high priority flight safety messages" for application service priority and "low" for RER quality of service. If these values are not present (as checked in sections 2.2.1.5.4.8 and 2.2.2.5.4.7 of the SARPs), then the ADS ASE will abort with reason "invalid-QOS-parameter".

3.2.10 Version Number Negotiation

3.2.10.1 The current version of the SARPs identifies the version number to be 1. In the future, it can be anticipated that further operational requirements will emerge, and therefore there is a need to develop further versions of the ADS SARPs.

3.2.10.2 Although this SARPs is not able to put any requirements on future versions of the SARPs, some consideration has been given for the future: It can be foreseen that there will be a time when there will be a mixture of aircraft and ground system ADS applications in the world — some implemented to version 1 of the ADS SARPs, and some to version 2 (and possible version 3 and beyond).

3.2.10.3 When the CM application exchanges addresses, it also exchanges the version numbers of the applications that are running on the ground and in the aircraft. It is the responsibility of the ground system to ensure that it does not attempt to establish a contract with a system that is working on a different version of ADS.

3.2.10.4 When the ADS report forwarding application is run, the situation is somewhat different. There is no exchange of addresses and version numbers through CM. Therefore, the version number of the application is passed during initiation of the connection. The receiving application must check the version number, and ensure that it can work to the same version number that the sending application has. If it is working on a lower version number, it can refuse the connection and pass to the initiating system the version number that it is implemented to.

3.3 Functionality of Services

3.3.1 Introduction

3.3.1.1 This section describes first the information required by the ASEs from the ADS-users. Then, it considers ADS functions in turn and provides an overview of the data flow within the ASE which handles the service primitives.

3.3.2 **Concepts**

3.3.2.1 Users of the ADS service are termed ADS-ground-user and ADS-air-user or just ADS-user when it applies to both air and ground. The ADS-user represents the operational part of the ADS system. It is either the final end-user (e.g. a crew member or controller) or an automated system. The ADS-user that initiates a ADS air-ground or ground-ground service is termed the calling ADS-user or initiator. The ADS-user that the initiator is trying to contact is termed the called ADS-user or responder.

3.3.3 **ADS Service Parameters**

3.3.3.1 ***Aircraft Identifier***

3.3.3.1.1 This parameter contains a 24 bit aircraft address which uniquely identifies an aircraft. The information is always provided by the ADS-ground-user when invoking an ADS-demand-contract, ADS-event-contract or ADS-periodic-contract request. The information is provided to the dialogue service provider and is used to locate the airborne ADS application address.

3.3.3.2 ***Class of Communications Service***

3.3.3.2.1 This parameter contains the class of communications service required for the ADS contracts with a given aircraft. The information may be provided by the ADS-ground-user when invoking an ADS-demand-contract, ADS-event-contract or ADS-periodic-contract request. If the ADS-ground-user already has one or more contracts with the aircraft, the parameter is ignored. If there are no contracts in place with the aircraft, the information is provided to the dialogue service provider and is used to select the class of communications service that is provided. If there are no contracts in place with the aircraft and the parameter is not provided, the dialogue service provider selects the class of communications service to use.

3.3.3.3 ***Contract Details***

3.3.3.3.1 This parameter contains the details of either a demand, event or periodic contract. The information is provided by the ADS-ground-user when invoking an ADS-demand-contract, ADS-event-contract or ADS-periodic-contract request. The information is delivered, unchanged, to the ADS-air-user and is used to determine the conditions of the contract.

3.3.3.4 ***Contract Type***

3.3.3.4.1 This parameter contains an indication of the type of contract (demand, event or periodic). The information is provided by the ADS-air-user when invoking an ADS-report request, and by the ADS-ground-user when invoking an ADS-cancel-contract request (although in this latter case — the value of “demand contract” is not permitted). It is provided to the ADS-ground-user in the ADS-report indication for information purposes. It is provided to the ADS-air-user in the ADS-cancel-contract indication in order to indicate which contract to cancel.

3.3.3.5 *Emergency Report Details*

3.3.3.5.1 This parameter contains the emergency report that is provided as part of an emergency contract. The information is provided by the ADS-air-user when invoking an ADS-emergency-report request. It is provided to the ADS-ground-user for information.

3.3.3.6 *Event Type*

3.3.3.6.1 This parameter contains an indication of the type of event that has occurred in an event contract. The information is provided by the ADS-air-user when invoking an ADS-report request for an event contract. (Note that an event contract can require reports on more than one type of event, some of which may report the same information. It is therefore necessary to report which type of event occurred.) The information is provided to the ADS-ground-user for information.

3.3.3.7 *ICAO Facility Designation*

3.3.3.7.1 This parameter contains the four to eight character ICAO facility designation of the ground system that is initiating the request. The information is provided by the ADS-ground-user user when invoking an ADS-demand-contract, ADS-event-contract or ADS-periodic-contract request, but only when the ground system has no contracts in place with the aircraft already. The information is provided to the ADS-air-user. If, at a later stage, the ADS-air-user has to reject an attempt to make a contract because its capacity is exceeded (e.g. it can only have contracts with four ground systems, and it already has contracts with four ground systems), then the ADS-air-user supplies this information to the ADS-ground-user when delivering its negative acknowledgement.

3.3.3.8 *Positive Acknowledgement*

3.3.3.8.1 This parameter contains no information — it is either present or absent. It may be provided by the ADS-air-user when invoking an ADS-report request. It is present if the ADS-air-user wishes to acknowledge an ADS-demand, ADS-event or ADS-periodic contract. It is absent otherwise. The information is provided to the ADS-ground-ASE as input to its state machine, and also to the ADS-ground-user for information.

3.3.3.9 *Positive Acknowledgement of Modification*

3.3.3.9.1 This parameter contains no information — it is either present or absent. It may be provided by the ADS-air-user when invoking an ADS-emergency-report request. It is present if the ADS-air-user wishes to acknowledge an ADS-emergency-contract-modification request. It is absent otherwise. The information is provided to the ADS-ground-ASE as input to its state machine, and also to the ADS-ground-user for information.

3.3.3.10 ***Reason***

3.3.3.10.1 This parameter indicates the reason for an abort. It is provided by an ADS ASE when invoking ADS-provider-abort indication. It is used by the ADS-user for information.

3.3.3.11 ***Reply***

3.3.3.11.1 This parameter contains a positive acknowledgement, a noncompliance notification or a negative acknowledgement. It is provided by the ADS-air-user when invoking an ADS-demand-contract, ADS-event-contract or ADS-periodic contract response. The information is provided to the ADS-ground-ASE as input to its state machine, and also to the ADS-ground-user for information.

3.3.3.12 ***Report Details***

3.3.3.12.1 This parameter contains the ADS report that is provided as part of a demand, event or periodic contract. The information is provided by the ADS-air-user when invoking an ADS-emergency-report request. It is provided to the ADS-ground-user for information.

3.3.3.13 ***Reporting Interval***

3.3.3.13.1 This parameter contains a requested reporting interval for an emergency contract. The information is provided by the ADS-ground-user when invoking an ADS-modify-emergency-contract request. It is used by the ADS-air-user to select a new interval for providing emergency reports.

3.3.4 **Demand Contract Service**

3.3.4.1 The ADS-demand-contract service is used to set up an ADS demand contract between the ADS-ground-user and the ADS-air-user. It is initiated by the ADS-ground-user.

- The ADS-demand-contract request is passed to the ground HI module, which examines it to see which module to pass it to. The request is passed to the ground DC module.
- The ground DC module generates an ADS-demand-contract-PDU and passes it to the ground LI module. It also starts timer t-DC-1 in order to monitor the time before a reply is received.
- The ground LI module decides how to use the dialogue service. If a dialogue already exists, it makes use of that dialogue. It uses the D-DATA service to pass the APDU to the airborne system. If no dialogue exists, it uses the D-START service to pass the APDU.
- The APDU is passed, via the upper layers and the network and emerges in the air LI module.
- The air LI module examines the APDU in order to determine which module to pass it to. The APDU is passed to the air DC module.

- The air DC module recognises the APDU as an ADS-demand-contract-PDU and passes an ADS-demand-contract indication to the air HI module.
- The air HI module passes it to the ADS-air-user.

3.3.4.2 The ADS-air-user then processes the ADS-demand-contract indication.

- The ADS air user examines the demand contract and determines one of the following:
  - if it can fulfil the request in its entirety, or
  - if it can obtain some of the information requested, but not all, or
  - if it cannot fulfil the request.
- It then replies (respectively) with:
  - an ADS-report request (including a positive acknowledgement parameter), or
  - an ADS-demand-contract indication (indicating noncompliance indicating the items of information that it is not able to supply), followed by an ADS-report request, or
  - an ADS-demand-contract indication (indicating negative acknowledgement).

3.3.4.3 The reply is then send to the ADS-ground-user.

- On receipt of an ADS-demand-contract response and/or ADS-report request, the ADS HI module decides where to pass it, and passes it to the air DC module.
- The air DC module then formulates the appropriate PDU (ADS-demand-report-PDU, ADS-noncompliance-notification-PDU, or ADS-negative-acknowledgement-PDU) and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA request (if a connection is already open) or a D-START response (if no D-START response has yet been given).
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground DC module.
- The ground DC module stops the t-DC-1 timer, decodes the PDU and generates an ADS-report indication, or an ADS-demand-contract confirmation (depending on which type of PDU it was passed). This is passed to the ground HI module.
- The ground HI module then passes the ADS-report indication or the ADS-demand-contract response to the ADS ground user.
- If a noncompliance notification was sent, the actions described in this section will occur twice, once for the ADS-noncompliance-notification-PDU, and once for the ADS-demand-report-PDU.

3.3.4.4 An ADS demand contract cannot be cancelled. A second ADS contract cannot be initiated until the first has responded either with a negative acknowledgement, or an ADS report.

3.3.4.4.1 It must be noted that the ADS demand contract service is different to the ADS periodic and event contracts in one major respect. In the later, it is possible to reply to the ADS contract request with a positive acknowledgement, and then later on provide the ADS report. This allows a periodic or event report to be set up over, say, a thirty second timeframe before the first report is sent. The demand contract, on the other hand, does not have this option.

3.3.4.4.2 The operational requirement for the demand contract is that the information is provided as soon as possible. Thus its intended use is for a ground system that requires “instant” information. Normal operational use of ADS is intended to be through periodic and event contract. The periodic contract, therefore, does not allow the possibility of delaying the provision of the ADS report. If some element that is requested from the report is not available immediately, then the ADS air user responds with a noncompliance notification, and send what information it does have immediately.

3.3.5 Event Contract Service

3.3.5.1 The ADS-event-contract service is used to set up an ADS event contract between the ADS-ground-user and the ADS-air-user. It is initiated by the ADS-ground-user.

- The ADS-event-contract request is passed to the ground HI module, which examines it to see which module to pass it to. The request is passed to the ground EC module.
- The ground EC module generates an ADS-event-contract-PDU and passes it to the ground LI module. It also starts timer t-EC-1 in order to monitor the time before a reply is received.
- The ground LI module decides how to use the dialogue service. If a dialogue already exists, it makes use of that dialogue. It uses the D-DATA service to pass the APDU to the airborne system. If no dialogue exists, it uses the D-START service to pass the APDU.
- The APDU is passed, via the upper layers and the network and emerges in the air LI module.
- The air LI module examines the APDU in order to determine which module to pass it to. The APDU is passed to the air EC module.
- The air EC module recognises the APDU as an ADS-event-contract-PDU and passes an ADS-event-contract indication to the air HI module.
- The air HI module passes it to the ADS-air-user.

3.3.5.2 The ADS-air-user then processes the ADS-event-contract indication.

- The ADS air user examines the event contract and determines one of the following:
  - if it can fulfil the request in its entirety, or
  - if it can fulfil the request in its entirety, but cannot respond with the first report within half a second, or
  - if it can obtain some of the information requested, but not all, or
  - if it cannot fulfil the request.
- It then replies (respectively) with:
  - an ADS-report request (including a positive acknowledgement parameter), or
  - an ADS-event-contract indication (indicating positive acknowledgement), followed by an ADS-report request when the information is available, or
  - an ADS-event-contract indication (indicating noncompliance), followed by an ADS-report request indicating the items of information that it is not able to supply, or
  - an ADS-event-contract indication (indicating negative acknowledgement).

3.3.5.3 The reply is then send to the ADS-ground-user.

- On receipt of an ADS-event-contract response and/or ADS-report request, the ADS HI module decides where to pass it, and passes it to the air EC module.
- The air EC module then stops the t-EC-1 timer, formulates the appropriate PDU (ADS-event-report-PDU, ADS-positive-acknowledgement-PDU, ADS-noncompliance-notification-PDU, or ADS-negative-acknowledgement-PDU) and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA request (if a connection is already open) or a D-START response (if no D-START response has yet been given).
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground EC module.
- The ground EC module decodes the PDU and generates an ADS-report indication, or an ADS-event-contract confirmation (depending on which type of PDU it was passed). This is passed to the ground HI module.
- The ground HI module then passes the ADS-report indication or the ADS-event-contract response to the ADS ground user.
- If a noncompliance notification or a positive acknowledgement was sent, the actions described in this section will occur twice, once for the ADS-noncompliance-notification-PDU or ADS-positive-acknowledgement-PDU, and once for the ADS-event-report-PDU.

3.3.5.4 The ADS-air-user then continues to fulfil the terms of the ADS contract by sending ADS-reports when one of the specified events occur.

- On receipt of an ADS-report request, the ADS HI module decides where to pass it, and passes it to the air PC module.
- The air EC module then formulates the ADS-periodic-report-PDU and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground EC module.
- The ground EC module decodes the PDU and generates an ADS-report indication. This is passed to the ground HI module.
- The ground HI module then passes the ADS-report indication to the ADS ground user.

3.3.5.5 At any time during the lifetime of the contract, the ADS ground user may cancel the event contract by invoking the ADS-cancel service.

- On receipt of an ADS-cancel request, the ground HI module decides where to pass it, and passes it to the ground EC module.

- The ground EC module then formulates the ADS-cancel-contract-PDU and passes it to the ground LI module.
- The ground EC module starts the t-EC-2 timer.
- The ground LI module then passes the PDU to the air using a D-DATA.
- The upper and lower layers then pass the PDU to the air ADS ASE.
- The air LI module examines the PDU and determines that it must be passed to the ground EC module.
- The air EC module decodes the PDU and generates an ADS-cancel indication. This is passed to the air HI module.
- The air HI module then passes the ADS-cancel indication to the ADS air user.
- The air EC module then creates an ADS-positive-acknowledgement-PDU and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground EC module.
- The ground EC module stops the t-EC-2 timer, decodes the PDU and generates an ADS-cancel confirmation. This is passed to the ground HI module.
- The ground HI module then passes the ADS-cancel confirmation to the ADS ground user.

3.3.6 Periodic Contract Service

3.3.6.1 The ADS-periodic-contract service is used to set up an ADS periodic contract between the ADS-ground-user and the ADS-air-user. It is initiated by the ADS-ground-user.

- The ADS-periodic-contract request is passed to the ground HI module, which examines it to see which module to pass it to. The request is passed to the ground PC module.
- The ground PC module generates an ADS-periodic-contract-PDU and passes it to the ground LI module. It also starts timer t-PC-1 in order to monitor the time before a reply is received.
- The ground LI module decides how to use the dialogue service. If a dialogue already exists, it makes use of that dialogue. It uses the D-DATA service to pass the APDU to the airborne system. If no dialogue exists, it uses the D-START service to pass the APDU.
- The APDU is passed, via the upper layers and the network and emerges in the air LI module.
- The air LI module examines the APDU in order to determine which module to pass it to. The APDU is passed to the air PC module.
- The air PC module recognises the APDU as an ADS-periodic-contract-PDU and passes an ADS-periodic-contract indication to the air HI module.
- The air HI module passes it to the ADS-air-user.

3.3.6.2 The ADS-air-user then processes the ADS-periodic-contract indication.

- The ADS air user examines the periodic contract and determines one of the following:

  - if it can fulfil the request in its entirety, or
  - if it can fulfil the request in its entirety, but cannot respond with the first report within half a second, or
  - if it can obtain some of the information requested, but not all, or
  - if it cannot fulfil the request.
- It then replies (respectively) with:
  - an ADS-report request (including a positive acknowledgement parameter), or
  - an ADS-periodic-contract indication (indicating positive acknowledgement), followed by an ADS-report request when the information is available, or
  - an ADS-periodic-contract indication (indicating noncompliance), followed by an ADS-report request indicating the items of information that it is not able to supply, or
  - an ADS-periodic-contract indication (indicating negative acknowledgement).

3.3.6.3 The reply is then sent to the ADS-ground-user.

- On receipt of an ADS-periodic-contract response and/or ADS-report request, the ADS HI module decides where to pass it, and passes it to the air PC module.
- The air PC module then formulates the appropriate PDU (ADS-periodic-report-PDU, ADS-positive-acknowledgement-PDU, ADS-noncompliance-notification-PDU, or ADS-negative-acknowledgement-PDU) and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA request (if a connection is already open) or a D-START response (if no D-START response has yet been given).
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground PC module.
- The ground PC module stops the t-PC-1 timer, decodes the PDU and generates an ADS-report indication, or an ADS-periodic-contract confirmation (depending on which type of PDU it was passed). This is passed to the ground HI module.
- The ground PC module starts the t-PC-2 timer.
- The ground HI module then passes the ADS-report indication or the ADS-periodic-contract response to the ADS ground user.
- If a noncompliance notification or a positive acknowledgement was sent, the actions described in this section will occur twice, once for the ADS-noncompliance-notification-PDU or ADS-positive-acknowledgement-PDU, and once for the ADS-periodic-report-PDU.
- The ADS-air-user then continues to fulfil the terms of the ADS contract by sending ADS-reports at regular intervals.
- On receipt of an ADS-report request, the ADS HI module decides where to pass it, and passes it to the air PC module.
- The air PC module then formulates the ADS-periodic-report-PDU and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground PC module.

- The ground PC module decodes the PDU and generates an ADS-report indication. This is passed to the ground HI module; while doing this is stops the t-PC-2 timer, resets it, and starts it again.
- The ground HI module then passes the ADS-report indication to the ADS ground user.

3.3.6.4 At any time during the lifetime of the contract, the ADS ground user may cancel the periodic contract by invoking the ADS-cancel service.

- On receipt of an ADS-cancel request, the HI module decides where to pass it, and passes it to the ground PC module.
- The ground PC module then formulates the ADS-cancel-contract-PDU and passes it to the ground LI module.
- The ground PC module starts the t-PC-3 timer.
- The ground LI module then passes the PDU to the air using a D-DATA.
- The upper and lower layers then pass the PDU to the air ADS ASE.
- The air LI module examines the PDU and determines that it must be passed to the ground PC module.
- The air PC module decodes the PDU and generates an ADS-cancel indication. This is passed to the air HI module.
- The air HI module then passes the ADS-cancel indication to the ADS air user.
- The air PC module then creates an ADS-positive-acknowledgement-PDU and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground PC module.
- The ground PC module stops the t-PC-3 timer, decodes the PDU and generates an ADS-cancel confirmation. This is passed to the ground HI module.
- The ground HI module then passes the ADS-cancel confirmation to the ADS ground user.

3.3.7 Emergency Contract Service

3.3.7.1 While the ADS air user has ADS contracts in place, it may choose to initiate an emergency contract. This will be initiated either by one of the aircrew, or the avionics on detection of a problem. The ADS air user invokes ADS-emergency-report requests at regular intervals.

- On receipt of an ADS-emergency-report request, the ADS HI module decides where to pass it, and passes it to the air EM module.
- The air EM module then formulates the ADS-emergency-report-PDU and passes it to the air LI module. It also requests the PC module to suspend operation of its periodic contract, if it has one in place.
- The air LI module then passes the PDU to the ground using a D-DATA.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground EM module.

- The ground EM module decodes the PDU and generates an ADS-emergency-report indication. This is passed to the ground HI module. It also requests the PC module to suspend operation of its periodic contract, if it has one in place.
- If the t-EM-1 timer is running, the ground EM module stops it, resets it and restarts it.
- The ground HI module then passes the ADS-emergency-report indication to the ADS ground user.
- This action is repeated at regular intervals. (The EM modules only request the PC modules to suspend operation for the first time through.)

3.3.7.2 The ADS-modify-emergency-contract service is used to change the reporting interval for emergency contracts. It is initiated by the ADS-ground-user.

- The ADS-modify-emergency-contract request is passed to the ground HI module, which examines it to see which module to pass it to. The request is passed to the ground EM module.
- The ground EM module generates an ADS-modify-emergency-contract-PDU and passes it to the ground LI module. It also starts timer t-EM-2 in order to monitor the time before a reply is received.
- The ground LI module uses the D-DATA service to pass the APDU to the airborne system.
- The APDU is passed, via the upper layers and the network and emerges in the air LI module.
- The air LI module examines the APDU in order to determine which module to pass it to. The APDU is passed to the air EM module.
- The air EM module recognises the APDU as an ADS-modify-emergency-contract-PDU and passes an ADS-modify-emergency-contract indication to the air HI module.
- The air HI module passes it to the ADS-air-user.

3.3.7.3 The ADS-air-user then processes the ADS-modify-emergency-contract indication.

- The ADS air user examines the ADS-modify-emergency-contract indication and determines one of the following:
  - if it can change to the request reporting rate, or
  - if it cannot fulfil the request.
- It then replies (respectively) with:
  - an ADS-emergency-report request (including a positive acknowledgement parameter), or
  - an ADS-modify-emergency-contract response (indicating negative acknowledgement).

3.3.7.4 The reply is then sent to the ADS-ground-user.

- On receipt of an ADS-modify-emergency-contract response and/or ADS-emergency-report request, the ADS HI module decides where to pass it, and passes it to the air EM module.

- The air EM module then formulates the appropriate PDU (ADS-emergency-report-PDU or ADS-negative-acknowledgement-PDU) and passes it to the air LI module.
- The air LI module then passes the PDU to the ground using a D-DATA request.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground EM module.
- The ground EM module stops the t-EM-2 timer, decodes the PDU and generates an ADS-emergency-report indication, or an ADS-modify-emergency-contract confirmation (depending on which type of PDU it was passed). This is passed to the ground HI module.
- The ground HI module then passes the ADS-emergency-report indication or the ADS-modify-emergency-contract confirmation to the ADS ground user.

3.3.7.5 At any time during the lifetime of the emergency contract, the ADS air user may cancel the emergency contract by invoking the ADS-cancel-emergency service.

- On receipt of an ADS-cancel-emergency request, the HI module decides where to pass it, and passes it to the air EM module.
- The air EM module then starts the t-EM-3 timer, formulates the ADS-cancel-emergency-PDU and passes it to the air LI module.
- The air LI module then passes the PDU to the air using a D-DATA.
- The upper and lower layers then pass the PDU to the ground ADS ASE.
- The ground LI module examines the PDU and determines that it must be passed to the ground EM module.
- The ground EM module stops the t-EM-1 timer, decodes the PDU and generates an ADS-cancel-emergency indication. This is passed to the ground HI module.
- The ground HI module then passes the ADS-cancel-emergency indication to the ADS ground user.
- The ground EM module then creates an ADS-positive-acknowledgement-PDU and passes it to the ground LI module.
- The ground LI module then passes the PDU to the air using a D-DATA.
- The upper and lower layers then pass the PDU to the air ADS ASE.
- The air LI module examines the PDU and determines that it must be passed to the air EM module.
- The air EM module stops the t-EM-3 timer, decodes the PDU and generates an ADS-cancel-emergency confirmation. This is passed to the air HI module.
- The air HI module then passes the ADS-cancel-emergency confirmation to the ADS air user.

3.3.8 Abort Service

3.3.8.1 At any time while there are contracts in place, the ADS ground user or the ADS air user may abort the connection by invoking the ADS-user-abort service. The following description describes the action being initiated by the ADS ground user. The ADS air user can ipso facto initiate the sequence of events.

- On receipt of an ADS-user-abort request, the HI module decides where to pass it, and passes it to the ground AB module.
- The ground AB module then requests the ground LI module to abort. It also requests all other modules to cease operation.
- The ground LI module invokes D-U-ABORT request.
- The upper and lower layers then pass the request to the air ADS ASE.
- The air LI module examines the PDU and determines that it must be passed to the air AB module.
- The air AB module generates an ADS-user-abort indication. This is passed to the air HI module. It also requests all other modules to cease operation.
- The air HI module then passes the ADS-user-abort indication to the ADS air user.

3.3.8.2 If any of the modules within the air ASE or the ground ASE detects a problem, either in itself, or in the response it has got from its peer, it can abort the connection. The following description describes the action being initiated by the ADS ground ASE. The ADS air ASE can ipso facto initiate the sequence of events.

- The module detecting the error requests the ground AB module to abort.
- The ground AB module then creates an ADS-provider-abort-PDU, and passes it to the ground LI module. It also creates an ADS-provider-abort indication and passes it to the ground HI module. It requests all other modules to cease operation.
- The ground HI module invokes the ADS-provider-abort indication.
- The ground LI module invokes D-U-ABORT request.
- The upper and lower layers then pass the request to the air ADS ASE.
- The air LI module examines the PDU and determines that it must be passed to the air AB module.
- The air AB module generates an ADS-provider-abort indication. This is passed to the air HI module. It also requests all other modules to cease operation.
- The air HI module then passes the ADS-provider-abort indication to the ADS air user.
- If the upper or lower layers detect an error, then they will generate a D-P-ABORT indication at both the ADS air ASE and the ADS ground ASE.
- The air and ground LI modules determine that the D-P-ABORT must be passed to the air AB module.
- The air and ground AB modules generate an ADS-provider-abort indication. These are passed to the air and ground HI modules. They also request all other modules to cease operation.
- The air and ground HI modules then pass the ADS-provider-abort indication to the ADS air and ground users.

- A full list of errors that generate an abort can be found in section 2.9.

3.3.9 Cancel All Contracts Service

3.3.9.1 The ADS-Cancel-all-contracts service is used by the ADS ground user to cancel all the contracts that the ground user has with the aircraft.

- On receipt of an ADS-cancel-all-contracts request, the HI module decides where to pass it, and passes it to the ground LI module.
- The ground LI module then creates an ADS-cancel-all-contracts-PDU and passes it to the air using a D-END.
- The upper and lower layers then pass the PDU to the air ADS ASE.
- The air LI module examines the PDU and passes an ADS-cancel-all-contracts indication to the air HI module.
- The air HI module then passes the ADS-cancel-all-contracts indication to the ADS air user.
- The air LI module then invokes D-END response.
- The upper and lower layers then pass the response to the ground ADS ASE.
- The ground LI module creates an ADS-cancel-all-contracts confirmation and passes it to the ground HI module.
- The ground HI module then passes the ADS-cancel-all-contracts confirmation to the ADS ground user.

3.3.10 **Automated Connection Closure**

3.3.10.1 After completion of any actions resulting from the arrival of a PDU from the aircraft, the ground LI module checks the status of all modules. If there are no contracts current, it closes the connection with the aircraft.

- The ground LI module invokes D-END request.
- The upper and lower layers then pass the request to the air ADS ASE.
- The air LI module invokes D-END response.
- The upper and lower layers then pass the request to the ground ADS ASE.

3.3.11 **ADS Report Forwarding**

3.3.11.1 The ADS-start-forward service is used to initiate the forwarding of ADS reports to another ground system.

- The ADS user invokes ADS-start-forward request.
- The ARF ASE in the initiating system creates an APDU and invokes D-START request.
- The ARF ASE in the receiving system receives the D-START indication checks the version number of the initiating system to see if it is compatible.
- If the version numbers are compatible, the ARF ASE in the receiving system then invokes ADS-start-forward indication and D-START response (accepting the connection).
- The ARF ASE in the initiating system receives the D-START confirmation and invokes ADS-start-forward confirmation.

3.3.11.2 The ADS-forward-report service is used to forward an ADS reports to another ground system. It is invoked by the initiating system, and may be invoked repeatedly once the connection is set up.

- The ADS user invokes ADS-forward-report request.
- The ARF ASE in the initiating system creates an APDU and invokes D-DATA request.
- The ARF ASE in the receiving system receives the D-DATA indication and invokes ADS-forward-report indication.3.3.11.3 The ADS-end-forward service is used to complete the forwarding of ADS reports to another ground system.
- The ADS user invokes ADS-end-forward request.
- The ARF ASE in the initiating system invokes D-END request.
- The ARF ASE in the receiving system receives the D-END indication, invokes ADS-end-forward indication, and invokes D-END response (accepting the closure).
- The ARF ASE in the initiating system receives the D-END confirmation.

3.4 ADS SARPs Section Description

3.4.1 SARPs Section 2.2.1.2 — General Requirements

3.4.1.1 *SARPs Section 2.2.1.2.1 — Version Number*

3.4.1.1.1 This section is included to allow the CM application to exchange version numbers of the ADS application. It is necessary to allow for future versions of the protocol to be negotiated by CM. It has no effect on the ADS functionality.

3.4.1.2 *SARPs Section 2.2.1.2.2 — Error Processing Requirements*

3.4.1.2.1 In the abstract service definition, each service has a set of parameters and the abstract syntax of those parameters specified. Thus information which is not a valid syntax is not allowed to be input.

3.4.1.2.2 In the protocol definition, there is a requirement that no service is permitted to be called when the ASE is in an inappropriate state. Thus making use of the abstract services is not permitted at these times.

3.4.1.2.3 An implementation would not normally allow the user to take such invalid actions; however, there is no requirement to prevent an implementation from allowing this. The error processing requirements section thus says that if the implementation allows the user to enter invalid information, the system must inform the user that an entry error has occurred.. In that case, the error is locally detected and the dialogue does not need to be aborted.

3.4.2 SARPs Section 2.2.1.3 — The Abstract Service

3.4.2.1 *The Concept of an Abstract Service*

3.4.2.1.1 Section 2.2.1.3 concerns the ADS abstract service. The following paragraphs provide an explanations of the term “Abstract Service”.

3.4.2.1.2 In order to define the ADS ASE (i.e. the part of the ADS abstract service provider that contains the protocol machine — see section 2.2.1.5 of the SARPs), it is necessary to describe its reactions to both PDUs arriving from the peer application, and the inputs from the user. The PDUs are well defined in the protocol. The actions of the user, however, are not. The SARPs do not attempt to dictate the actions of the user except where absolutely necessary. Despite this, in order to define the ASE it is necessary to have a clear definition of user actions.

3.4.2.1.3 In order to get around this conundrum, an "abstract service" is defined. An abstract service is a textual description of the interactions between the user and the ASE. These interactions are precisely defined in section 2.2.1.3 of the SARPs. Having this definition allows the ASE to be specified precisely in terms of its reactions to the arrival of PDUs and the invocation of the services by the user. This, therefore, is the reason for defining the abstract service.

3.4.2.1.4 The abstract service interface is defined as being an interface between the ASE and the "user-part" of the software. These are known as the ADS ASE and the ADS user. The ADS user is not generally the human user; it is that part of the system that uses the ADS ASE.

3.4.2.2 *The Concept of APIs*

3.4.2.2.1 If one was to buy an ADS application, one would be buying a suite of executable code. From the code itself it is impossible to know whether or not the abstract service interface has actually been implemented. Therefore the ADS SARPs do not require that the abstract service interface has to be built as a real interface. It only requires that, when one examines it from an external point of view, it behaves in the same way as if it had been built. This is the explanation of statement 2.2.1.3.1.1 in the SARPs.

3.4.2.2.2 Thus the implementors may choose to build an ADS application with a real internal interface that corresponds to the abstract service interface, or they may choose not to — it is entirely up to them. However, it must be realised that there are a number of good reasons why one might not want to build a system with an interface exactly like the abstract service interface. Examples include:

- There may be a more efficient way of building the software.
- The abstract service interface does not include parameters that are needed locally, but do not affect the state machine; for example, a real interface might include an indication of which aircraft an ADS report has come from.
- It may not be easy to build the abstract service from the development tools that are being used.
- The abstract service interface does not have any programming language bindings. A real interface would require an interface defined in a particular programming language.

3.4.2.2.3 Implementation of the abstract service interface is not mandated by the ADS SARPs. The requirements for ADS set out by ICAO are limited in scope — they are designed only to ensure interoperability between air and ground systems, and to ensure that they meet the stated functionality requirements. The ADS SARPs do not specify the nature of any internal interface within the software, nor do they specify the human interface. Individual implementation projects must define their own internal interfaces to suit their own requirements.

3.4.2.2.4 In summary, an abstract service interface is defined in the SARPs in order to be able to define the ASE protocol machine. It does not have to be built in any implementation. A real implementation of the ADS SARPs would normally be expected to define its own internal interfaces.

3.4.2.2.5 The ADS application abstract service consists of eleven functions listed in 2.2.1.3.2.1.

3.4.2.3 *Functional Model of the ADS Application*

3.4.2.3.1 Figure 2.2.1.3-1 shows an abstract model for the ADS application. This figure is duplicated in figure 3.4-1. Just as with the abstract service, this model shows a design of the ADS application, breaking it down into modules. However, there is no requirement that an implementation actually builds it this way. The figure is presented here in order to explain the terms that are used throughout the document. It is not required that the design of an implementation follows this structure.

3.4.2.3.2 The figure shows three modules:

- the ADS user (which could be an ADS air user or and ADS ground user);
- the control function;
- the ADS ASE (application service element — which could be an ADS air ASE or and ADS ground ASE).

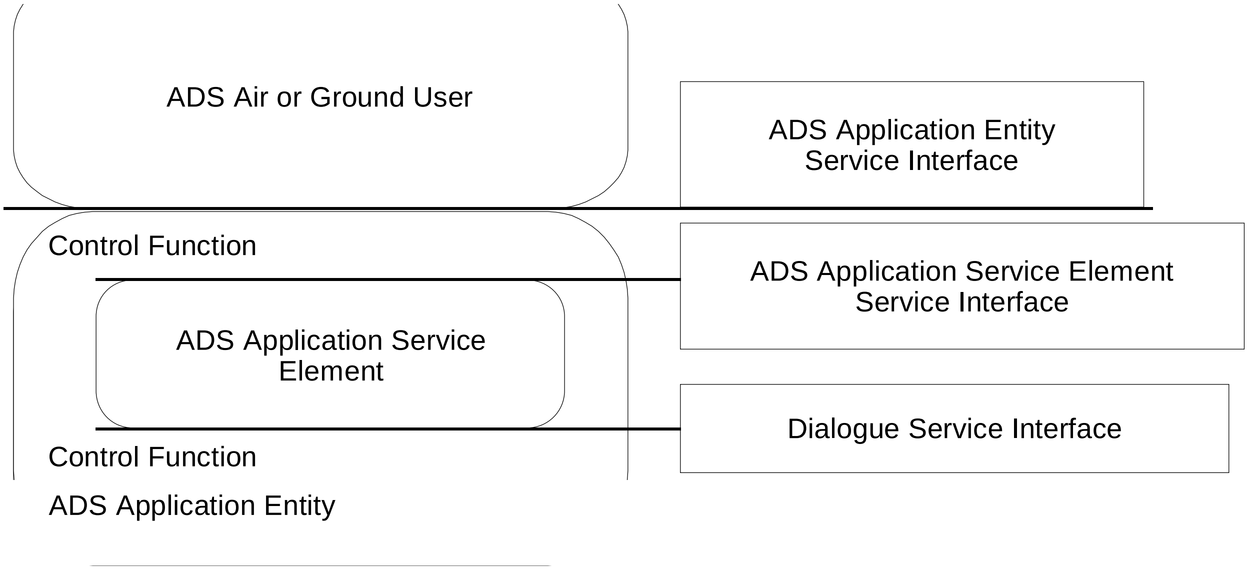

ADS Air or Ground User
ADS Application Entity Service Interface
Control Function
ADS Application Service Element Service Interface
ADS Application Service Element
Dialogue Service Interface
Control Function
ADS Application Entity

Figure 3.4-1. Abstract Model for the ADS Application

3.4.2.3.3 In addition, it defines the ADS application entity as the control function together with the ADS ASE.

3.4.2.3.4 Abstract interfaces are shown between the different modules:

- the ADS application entity service interface — which is the same as the abstract service interface defined in 2.2.1.3;

- the ADS application service element service interface — which is also the same as the abstract service interface defined in 2.2.1.3;

- the dialogue service interface — which is defined in the upper layer architecture SARPs, and is identical for all air/ground applications.

3.4.2.3.5 Since the ADS application entity service interface is identical with the ADS application service element service interface, the control function module passes primitive calls directly from one to the other without interference.

3.4.2.4 ***SARPs Section 2.2.1.3.3 — Conventions***

3.4.2.4.1 Service Primitives

3.4.2.4.1.1 The ADS SARPs defines 14 services (ADS-demand-contract, ADS-event-contract, ADS-periodic-contract, ADS-report, ADS-cancel, ADS-cancel-all-contracts, ADS-emergency-report, ADS-modify-emergency-contract, ADS-cancel-emergency-contract, ADS-user-abort, ADS-provider-abort, ADS-start-forward, ADS-report-forward and ADS-end-forward). Each primitive consists of the name of the ADS service and a suffix that indicates at what point in the service the primitive occurs (request, indication, response, confirmation), e.g. ADS-demand-contract request. The primitives are further explained below:

- the ADS user that initiates the service calls on the ADS ASE to perform an action — this is called the "request";
- after the request is passed to the ADS ASE on the other side of the communications link, it uses the service to pass the information on to its ADS user — this is called the "indication";
- the ADS user that has received the indication may choose to respond to it, in which case it calls upon its ADS ASE to send a reply — this is called the "response";
- finally, the ADS ASE receiving the response provides its ADS user (which started the sequence of events) with the information — this is called the "confirmation".

3.4.2.4.1.2 The terms "request", "indication", "response" and "confirmation" are well understood in the field of communications protocols. A given ADS service need not use all four primitives. Some ADS services make use of one (indication), some two (request and indication or request and confirmation), some three (request, indication and confirmation) and some all four (request, indication, response and confirmation). These primitives are illustrated in Figure 3.4-2.

3.4.2.4.1.3 A *confirmed ADS service* is one that involves a handshake between the user that requests the service and the user, or ASE, that is informed that the service has been requested. Figure 3.4-2 and Figure 3.4-3 illustrate the confirmed ADS services.

3.4.2.4.1.4 An *unconfirmed ADS service* involves no handshake. Figure 3.4-4 illustrates an unconfirmed service.

3.4.2.4.1.5 For a provider-initiated service, an indication primitive is given to both ADS-users. The provider-initiated service is generated by the service provider in response to an internal condition. Figure 3.4-5 illustrates a provider-initiated service.

3.4.2.4.2 Detailed Service Descriptions

3.4.2.4.2.1 Each of the detailed service descriptions is defined in the same way. Firstly there is either one or two tables indicating the parameters of the service and their status in each of the primitives. Secondly, each of the parameters has a short description, and a statement of what the abstract syntax of the parameter is. This is further described in the following sections.

3.4.2.4.3 Service Primitive and Parameters

3.4.2.4.3.1 Each service has a set of parameters. (One may choose to think of the service as a procedure or function calls in a programming language.) The parameters used in the request, indication, response and confirmation are different. (Thus one may choose to think of them as four different, but related, procedure calls.)

3.4. 2.4.
3.2 The
serv ices
are depi
cted i n
the S A
RPs b y
pri miti
v e and
par ame
ter tabl
es. A n
exa mpl
e of suc
h a tabl
e is sho
w n i n
Tab l e

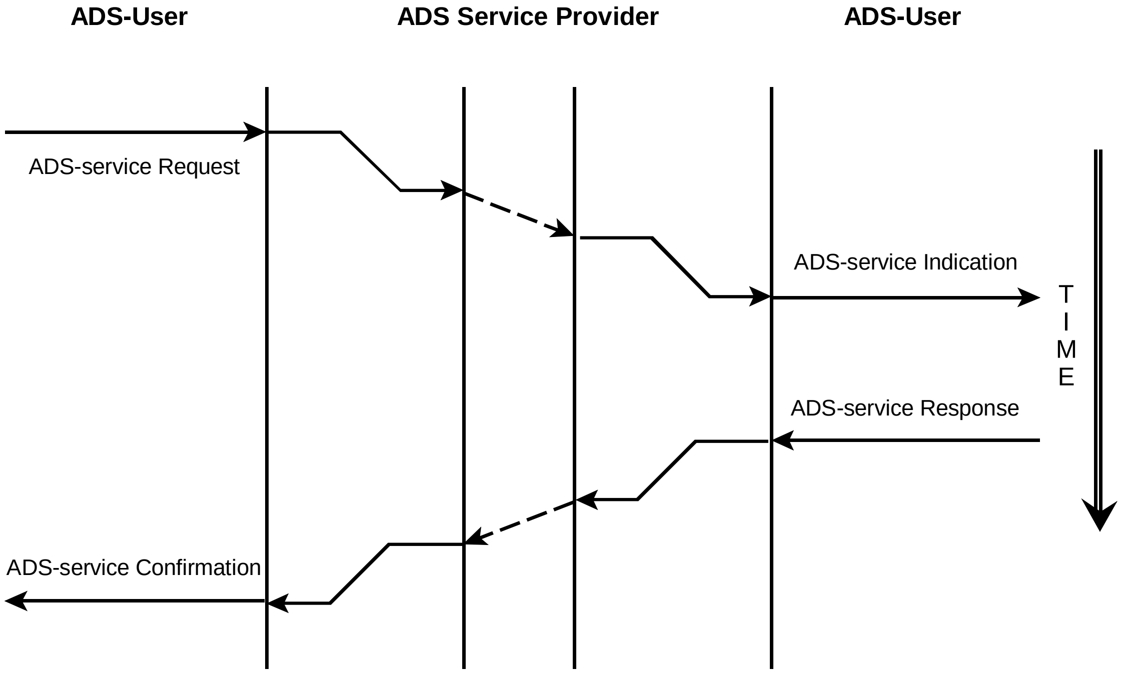


Figure 3.4-2. Generic ADS-User Primitives

3.4-1. Not all services require all primitives to be used. That is, if a particular primitive is not needed due to reasons like redundant information being relayed, then the parameter column for that primitive is omitted. If a parameter column for a primitive is present, but all of the parameters are left blank, that means that the primitive is used by the service but does not carry any data.

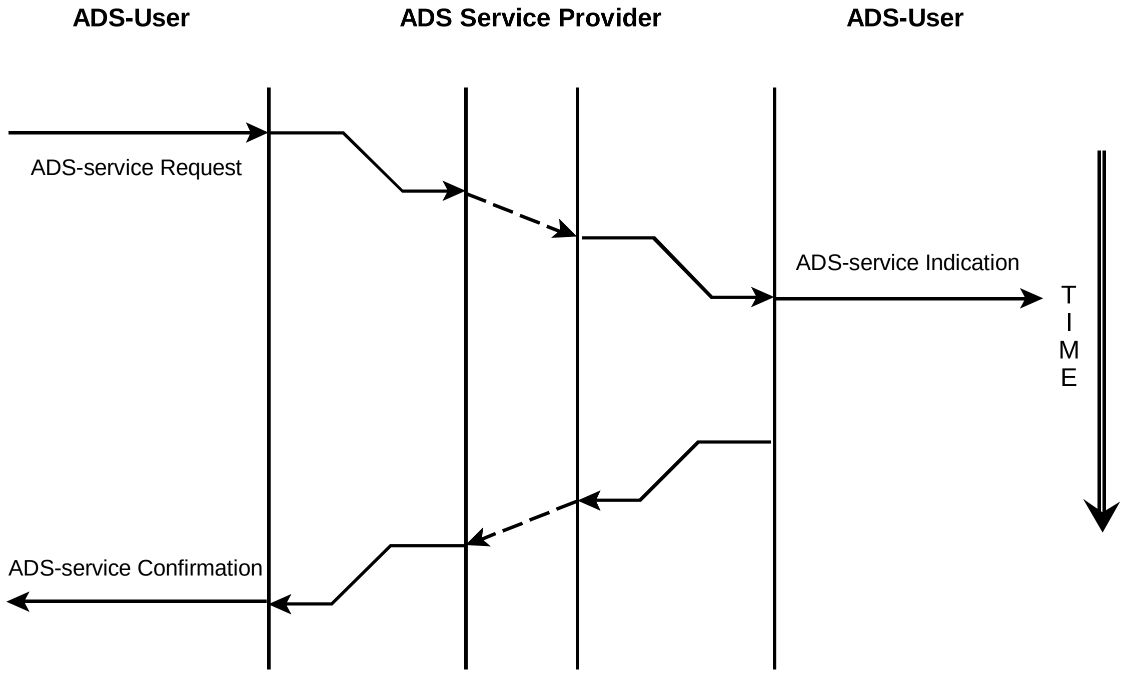


Figure 3.4-3. ???

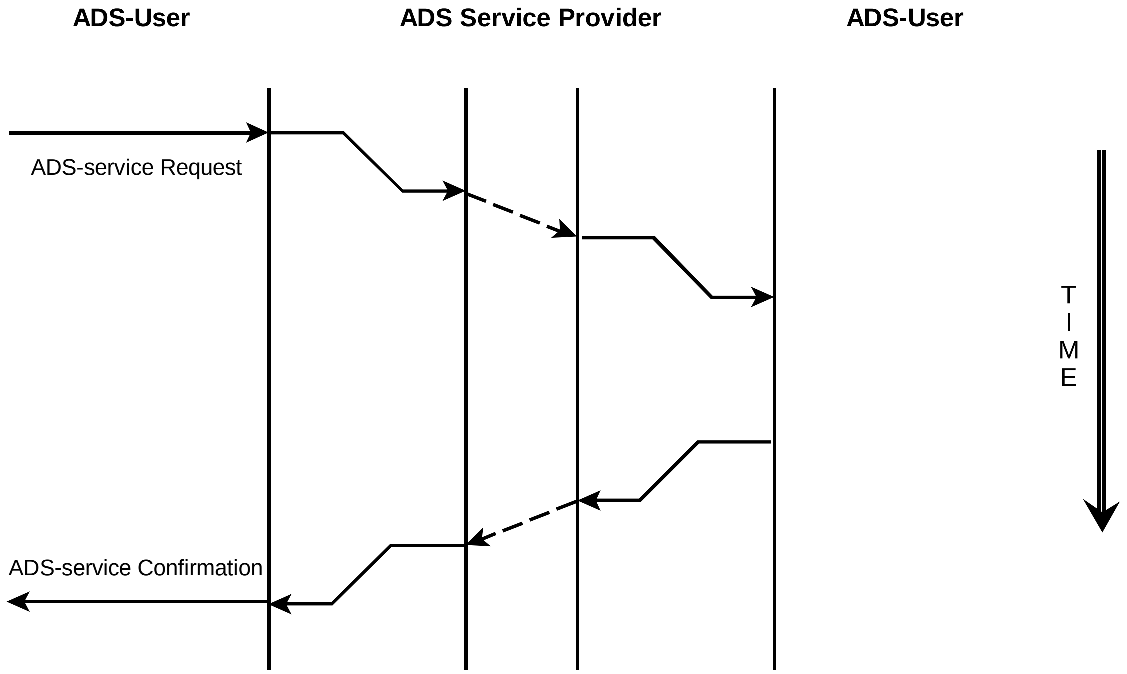


Figure 3.4-4. Other confirmed ADS services

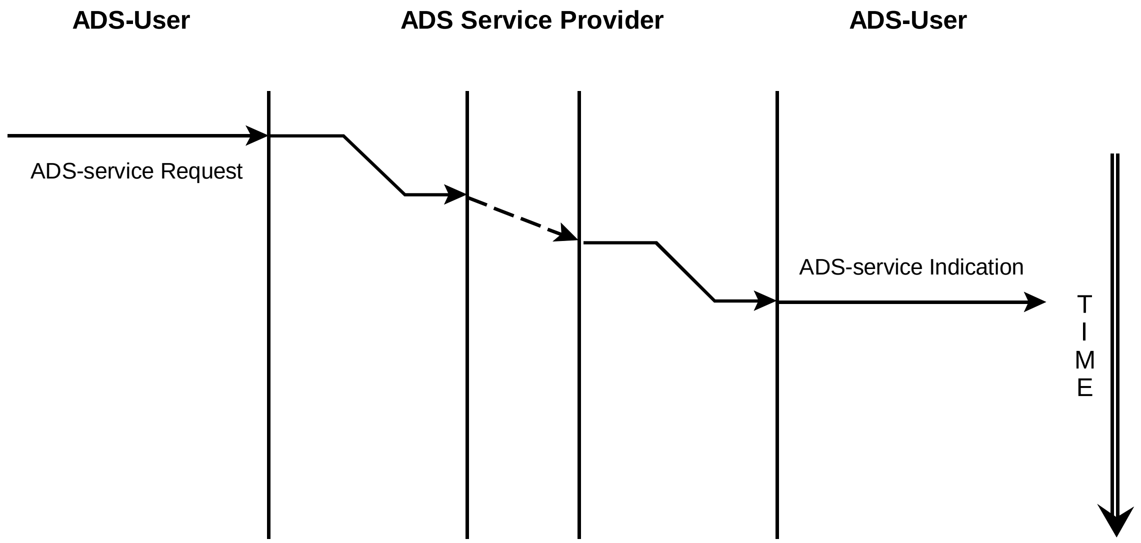


Figure 3.4-5. Generic unconfirmed ADS service

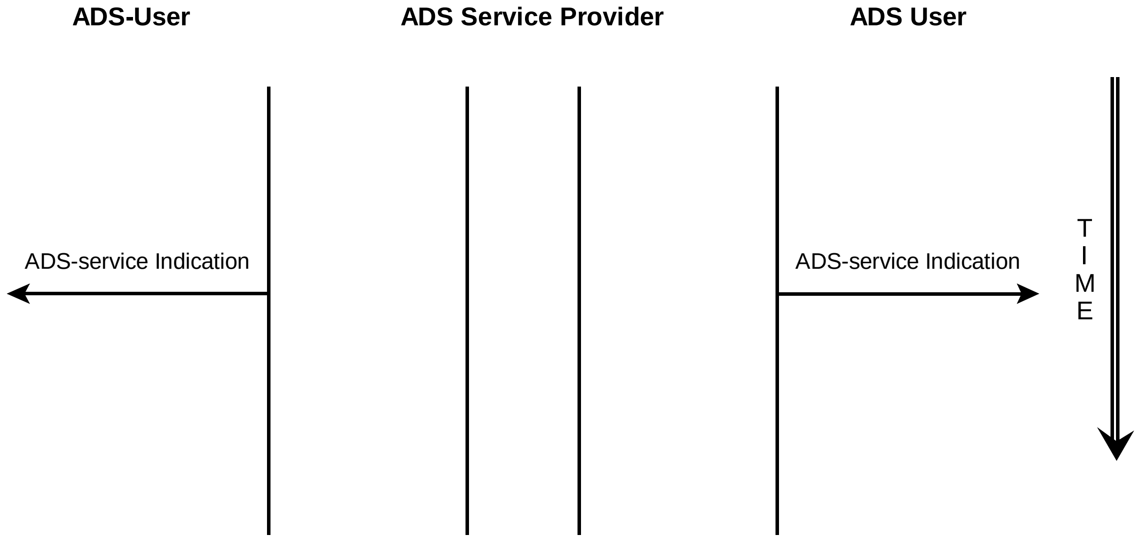


Figure 3.4-6. ADS service provider-initiated service

Table 3.4-1. Description of the ADS Service Primitives

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

3.4.2.4.3.3 For a specific primitive, each parameter is described by a value that dictates the terms under which that parameter is used. If the use of any parameter does not follow the rules as set forth by the primitive and parameter tables, there is an error in the implementation. The abbreviations used in the primitive and parameter sections are described below:

- blank — this means that the specific parameter will not be used in this service primitive.

- C — this means that the parameter is conditional upon some state. A “C” differs from a “U” (User Option) due to the fact that if the stated condition exists, the parameter must be supplied, while a “U” means that the parameter’s use is wholly up to the user. The exact conditions under which the parameter is used is explained in the text.

- C(=) — typically this is used when a request has an optional parameter. The C(=) will be in the “indication” position of the table. It means that if the user provides the parameter in the request, then it must also be present in the indication — what is more, it must have the same value as the parameter in the “request” column. The same applies to the “response” and “confirmation” columns.

- M — this means that the parameter must always be present, and no option not to use it exists.

- M(=) — this means that the parameter must always be present. If the parameter is in the “indication” column, then it must have a value equal to that of the “request” column. The same applies to the “response” and “confirmation” columns.

- U — this means that the use of the parameter is a user option. Therefore the presence of the parameter is optional, and will be used based upon user requirements. Some of the user requirements for “U” parameters are specified in the ADS SARPs 2.2.1.7 and 2.2.2.7. The use of “U” parameters are not wholly specified in the SARPs. Further requirements determining the use of “U” parameters may be determined by operational or procedural requirements outside the scope of the ADS SARPs.

3.4.2.4.4 When there are no parameters specified in the table (e.g. ADS-user-abort), this means that the service has the primitives listed in the table, but no parameters are needed.

3.4.2.4.5 A number of the services are presented with two parameter tables (ADS-demand-contract, ADS-event-contract and ADS-periodic-contract). The reason for this is that they may be operated in one of two ways. They may be operated with a request, indication, response and confirmation, or they may be operated with a request and indication only. Both cases are shown for these services.

3.4.2.4.6 The services ADS-demand-contract, ADS-event-contract and ADS-periodic-contract each has an optional parameter called "Class of communications service". Each of these services may all be called when a connection is in place, or when one is not in place. If the connection is not in place, then this parameter is used to help set up the connection. If the connection is in place already, then this parameter is not needed.

3.4.2.4.7 The class of communications service is defined as an optional parameter, therefore it may be provided when the connection is in place, or may be omitted when the connection is not in place. The behaviour defined in these cases is: if provided when a connection is in place it is ignored; if not provided when a connection is not already in place the SARPs states that "there is no routing preference". In practice this means that the implementors are free to choose what to do, for example, a default value could be chosen, a random value could be chosen or something else.

3.4.2.4.8 Throughout these service descriptions every parameter is described. In particular, there is a note that explains the purpose of the parameter, and a mandatory statement that states what values it may contain. In many cases, the mandatory statement states that if must conform to a named ASN.1 abstract syntax. These can be found in section 2.2.1.4.

3.4.2.5 *Service Parameters*

3.4.2.5.1 Throughout these service descriptions every parameter is described. In particular, there is a note that explains the purpose of the parameter, and a requirement that states what values it may contain.

3.4.2.5.2 Some primitive parameters have the same contents as APDU fields in the ASN.1 description. In most cases, the ADS-ASE is tasked to copy the parameter value within the APDU field. In order to avoid confusion by defining identical data structures twice, the type of those primitive parameters is specified by simply referring to the corresponding ASN.1 type in the APDU. The ASN.1 is used in the service definition as a syntax notation only and does not implicitly imply any local encoding of these parameters. The implementation of these parameters remains a local implementation issue.

3.4.2.5.3 For the other parameters, the syntax is described by enumerating the authorised abstract values.

3.4.2.6 *ADS Services*

3.4.2.6.1 This section lists the ADS services, and explains why each is a confirmed or an unconfirmed service.

3.4.2.6.2 ADS-demand-contract, ADS-event-contract and ADS-periodic-contract can be either confirmed or unconfirmed services. An ADS-air-user may reply to any one of these services by sending an ADS-report immediately. In such cases, the ADS contract service is unconfirmed. The ADS-report effectively carries the confirmation with it in the acknowledgement parameter. If the ADS-air-user replies with a positive or negative acknowledgement or a noncompliance notification, then it is a confirmed service. These services need to be confirmed since the ADS-ground-user needs to know at what point the contract is in place in order that it can monitor the results. For example, if a periodic contract is in place and the ADS-ground-user changes the contract, then the receipt of the confirmation signifies the point at which the change of contract takes place.

3.4.2.6.3 The ADS-report service is unconfirmed, since there is no requirement for the airborne system to know when the ground system has received the report.

3.4.2.6.4 The ADS-cancel and ADS-cancel-all-contracts services are both confirmed. It is necessary to confirm the service since the cancellation may cross with an ADS-report being sent from the aircraft. The ground system must know when the cancellation has been received so that it knows when to expect a cessation of ADS-reports. Neither service has a response primitive; the response is generated automatically. This is because the ADS-air-user has no option but to accept a cancellation.

3.4.2.6.5 The ADS-emergency-report service is unconfirmed, since there is no requirement for the airborne system to know when the ground system has received the report.

3.4.2.6.6 The ADS-modify-emergency-contract service is sometimes a confirmed service and sometimes an unconfirmed service. When the modification is accepted by the airborne system, an acknowledgement is sent down with the next ADS-emergency-report — this acts as the confirmation. When it is rejected, the airborne system confirms the service.

3.4.2.6.7 The ADS-cancel-emergency service is an unconfirmed service since there is no requirement for the airborne system to know when the ADS-cancel-emergency arrived at the ground.

3.4.2.6.8 The ADS-user-abort service is an unconfirmed service since there is no requirement for the initiating system to know that the abort has arrived at the peer.

3.4.3 SARPs Section 2.2.1.4 — Formal Definition of Messages

3.4.3.1 *Introduction*

3.4.3.1.1 The ADS abstract syntax 2.2.1.4 and 2.2.2.4 is written in a notation that is called ASN.1. It is strongly recommended that the reader is familiar with ASN.1 before attempting to understand the detail of these sections.

3.4.3.2 *ADS SARPs 2.2.1.4.1 Encoding/Decoding Rules*

3.4.3.2.1 This section defines which APDUs the systems must be able to encode and decode.

3.4.3.2.2 An APDU (Application Protocol Data Unit) is a sequence of bits that are passed from one peer application to another. The sequence of bits is a concrete representation of a message that is passed between applications. For example, the ASN.1 defines a message for an ADS-demand-contract. An APDU for this message will be a sequence of bits representing the information to be carried.

3.4.3.2.3 A system is not required to be able to encode or decode all messages specified in the ASN.1 description. An air system is not required to be able to encode an ADSGroundMessage APDU nor to decode an ADSAircraftMessage APDU.

3.4.3.3 *ADS ASN.1 Abstract Syntax*

3.4.3.3.1 SARPs sections 2.2.1.4 and 2.2.2.4 define the abstract syntax of the protocol. That is, it defines the structure of the PDUs that are to be sent between aircraft and the ground systems. It is written in a notation that is called ASN.1. It is strongly recommended that the reader is familiar with ASN.1 before attempting to understand the detail of this SARPs section.

3.4.3.3.2 Data types exchanged by ADS ASEs are described in the ADS SARPs by using a machine-independent and language-independent syntax. There is no constraint put on the implementors concerning the machine nor the development language to be selected for implementing the protocol.

3.4.3.3.3 The ASN.1 module MessageSetVersion1 contains the data types of the protocol data units handled by the ADS ASEs. Unlike common OSI ASEs (e.g. ACSE), no object identifier has been attached to the ADS ASN.1 specification. Indeed, the ULCS architecture releases the applications from negotiating during the dialogue establishment the applicable abstract syntax. Object identifiers related to ADS applications (application context name and version number) are defined in the ULCS SARPs.

3.4.3.4 *ADS ASN.1 Organisation*

3.4.3.4.1 This section defines the abstract syntax of the protocol. That is it defines the structure of the PDUs that are to be sent between aircraft and the ground systems. It is written in a notation that is called ASN.1. It is strongly recommended that the reader is familiar with ASN.1 in order to understand the details of section 2.2.1.4.2.

3.4.3.4.2 The ASN.1 itself is organised into a number of different sections:

- Aircraft generated and Ground generated message choice
- Ground generated and aircraft generated message components — protocol data units

- Reports and their components
- Components of contracts
- Miscellaneous components
- Common components
- ADS Report Forwarding Application

3.4.3.4.3 The top level (which will typically be used as an entry point by any ASN.1 compiler), is titled “Aircraft generated and Ground generated message choice”. It consists of two structures: one is a choice of PDUs generated by the aircraft, and the other is a choice of PDUs generated by the ground system.

3.4.3.4.4 The ASN.1 is written in such a way that every type is defined after it is used. This is sometimes a requirement in an ASN.1 compiler.

3.4.3.4.5 There are some components of reports that contain data from the aircraft sensors, for example, temperature, which is a component of weather. Comments that explain the units that the value is measured in and the range of values follow each of these. So that temperature, for example, can range from minus 100 degrees to plus 100 degrees Centigrade in steps of a quarter of a degree.

3.4.3.4.6 An index of the types defined in the ASN.1 is given in section ASN.1 Index.

3.4.3.5 *ASN.1 Tags*

3.4.3.5.1 Tags are used in ASN.1 to allow to distinguish data types when confusion is possible. For instance, when a data type contains two optional elements of the same type, if only one is encoded then there is no means for the decoder to know which element the decoded value is attached to.

3.4.3.5.2 Even if tag values are not used by the Packed Encoding Rules, the ASN.1 grammar mandates the use of tags in some cases. When specifying the CM data types, the following rules have been used:

- tags are always used within CHOICE data type, starting at 0 and then incremented by 1 for each entry;
- tags are not used at all in SEQUENCE data type when no confusion is possible. When an optional element is defined, all elements in the sequence are tagged.

3.4.3.6 *Extensibility Markers*

3.4.3.6.1 In order to allow the upgrade of the ASN.1 specification when new requirements are determined, the extensibility ASN.1 feature (ellipse, depicted by "...") has been used in applicable data types, for example, AbortReason. This allows future modifications to data types as the applications evolve while retaining backwards compatibility.

3.4.3.7 *Handling Numerical Values*

3.4.3.7.1 Some operational data are real. REAL data types exist in ASN.1, but the associated encoding procedures are not optimised in terms of data size (Packed Encoding Rules do not specify specific rules for REAL, but import the Basic Encoding Rules ones which require the encoding of a mantissa, a base and an exponent taking several octets). Real values are therefore specified via an INTEGER data type.

3.4.3.7.2 ASN.1 constraints are defined for INTEGER data types to take advantage of both the range and resolution. The lower bound is equal to the operational minimal value devised by the resolution, and the upper bound is equal to the operational maximal value devised by the resolution. The resolutions is referred to in the comments of the ASN.1 as "units".

3.4.3.7.3 For instance, in the definition of Weather, temperature is defined as:

```
temperature        [2]    INTEGER (-400..400) OPTIONAL,
                          -- units = 0.25 degrees Centigrade
                          -- range = -100 to 100 degrees C
```

3.4.3.7.4 This definition of real fields allows for a very simple algorithm for computing the operational value from the encoded value and vice-versa. The message sender divides the operational value (e.g. -12.5 degrees Centigrade) by the resolution (0.25) to find the value to send (-50) whereas the receiver multiplies the received value by the resolution to find the operational value.

3.4.3.8 *Entry Points*

3.4.3.8.1 The top level (which will typically be used as an entry point in any ASN.1 compiler), is titled "Aircraft generated and Ground generated message choice". It consists of two structures: one is a choice of PDUs generated by the aircraft, and the other is a choice of PDUs generated by the ground system.

3.4.3.8.2 An index of the types defined in the ASN.1 is given in section 6.

3.4.3.9 *Time Representation*

3.4.3.9.1 Data types have been specified for containing time indication (Date, DateTime, Year, Month, Hours, Minutes, Seconds). This way of representing time is preferred over the pre-defined ASN.1 representations (GeneralizedTime and UTCTime) for optimization of the PER encoding.

3.4.3.10 *Reason Code*

3.4.3.10.1 In an earlier version of the SARPs, there was a reason code called "cannot-meet-reporting-rate". This was put in to indicate, in response to a periodic contract, that the requested reporting rate could not be met. One of the changes in the SARPs that was made subsequently was that, if the aircraft could not meet the report rate, it delivered a default value reporting rate. During this change, the reason code "cannot-meet-reporting-rate" was not removed. When this defect was noticed, there were several implementations of ADS in existence. In order to avoid changing these implementations, the value for "cannot-meet-reporting-rate" was left in the ASN.1, but the name was changed to "undefined-reason" — thus allowing the old and new versions to interoperate.

3.4.3.11 *Unbounded ASN.1 Types*

3.4.3.11.1 There are a number of unbounded ASN.1 types in the definition of ADS APDUs. In practice, an implementation must have a maximum size for every type in order to reserve sufficient memory. The following paragraphs provide guidance on the maximum size that should be implemented.

3.4.3.11.2 The definition of Noncompliance Notification is:

```
NoncomplianceNotification ::= CHOICE
{
      demand-ncn          [0]     SET OF ReportType,
      event-ncn           [1]     SET OF EventTypeContracted,
      periodic-ncn        [2]     SET OF ReportTypeAndPeriod,
      ...
}
```

3.4.3.11.3 The three sets are unbounded. A valid implementation will not put more that seven ReportTypes in a NoncomplianceNotification for a demand-ncn, more than eleven EventTypeContracted in a NoncomplianceNotification for a event-ncn, or eight ReportTypeAndPeriod in a NoncomplianceNotification for a demand-ncn.

3.4.3.11.4 The definition of GroundSystemsUsingService is:

```
GroundSystemsUsingService ::= SEQUENCE OF Ia5String (SIZE(4..8))
```

3.4.3.11.5 This is used to inform a ground user that the air system capacity has been used up. e.g. if the implementation allows contracts with six ground systems, then when a seventh attempts to make a connection, GroundSystemsUsingService is used to inform the seventh ground user of the ICAO facility designations of the first six.

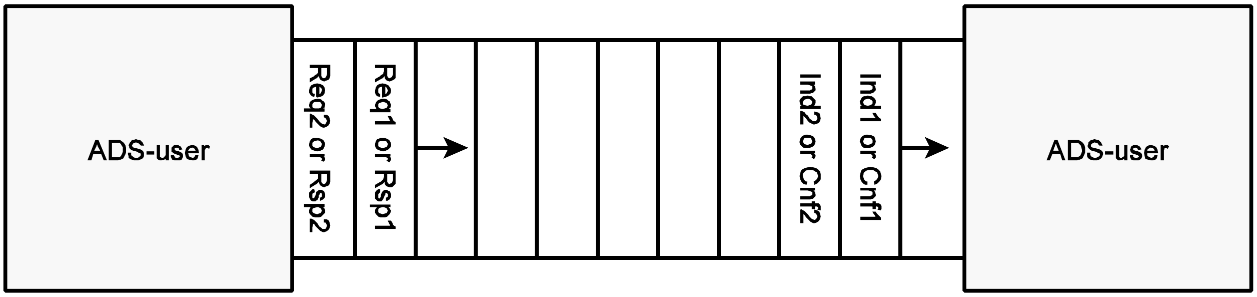


Figure 3.4-7. Communication queue

3.4.3.11.6 In an aircraft implementation, the maximum size for this type should be set by the maximum number of ground systems that the implementation can have contracts with. e.g. If the aircraft allows contracts with six ground systems, then this type should have a maximum size of six.

3.4.3.11.7 In a ground system implementation, some limit should be set. A size of, say, twenty is unlikely to be exceeded in any realistic implementation.

3.4.4 SARPs Section 2.2.1.5 Protocol Definition

3.4.4.1 *SARPs Section 2.2.1.5.1 Sequence Rules*

3.4.4.1.1 Time sequence diagrams or message sequence diagrams are used to denote the relationship between the primitives that form an ADS service and the order in which they occur, e.g. the indication/confirmation primitives occurs some time after the request/response primitives.

3.4.4.1.2 Inherent to the service model is the notion of queuing. The ADS-service indications and confirmations are delivered to the ADS-users in the order that the corresponding ADS-service requests and responses were issued. One exception to the notion of queuing is the abortive services (ADS-user-abort and ADS-provider-abort services) which may overtake other primitives and empty the primitives in the queue. Figure 3.4-5 illustrates this concept.

3.4.4.1.3 Each figure has the same structure. There are four vertical lines that separate the five major components in the ADS system. From left to right, they are:

- the ADS ground user — that part of the ground system that uses the ADS service to provide information to human users;

- the ADS ground ASE — that part of the ground system that implements the ADS protocol;

- the dialogue service provider — that part of the ground system, the air system and the networks that, together, provide the dialogue service, as defined in the upper layer architecture — on the figures this is the thin strip down the middle;

- the ADS air ASE — that part of the air system that implements the ADS protocol;

- the ADS air user — that part of the avionics that implements the ADS contracts.

3.4.4.1.4 The middle three sections of the diagrams (ADS ground ASE, dialogue service provider and ADS air ASE) together form the ADS service provider, and are labelled as such on the diagrams.

3.4.4.1.5 The outer two vertical lines represent the ADS abstract service. Any lines crossing them represent the invocation of one of the ADS service primitives. The ADS service primitives are labelled in the ADS user part of the figure.

3.4.4.1.6 The inner two vertical lines represent the Dialogue service. Any lines crossing them represent the invocation of one of the Dialogue service primitives. The Dialogue service primitives are labelled in the ADS ASE part of the figure.

3.4.4.1.7 The diagrams represent a sequence of events. Time is always considered to run down the figure from the top (representing the earliest time) to the bottom (representing the latest time).

3.4.4.1.8 If the ASEs set timers, these are marked on the figures by vertical lines with arrows at both ends. This is depicted in figure 3.4-6 by the "t_{timer}" label.

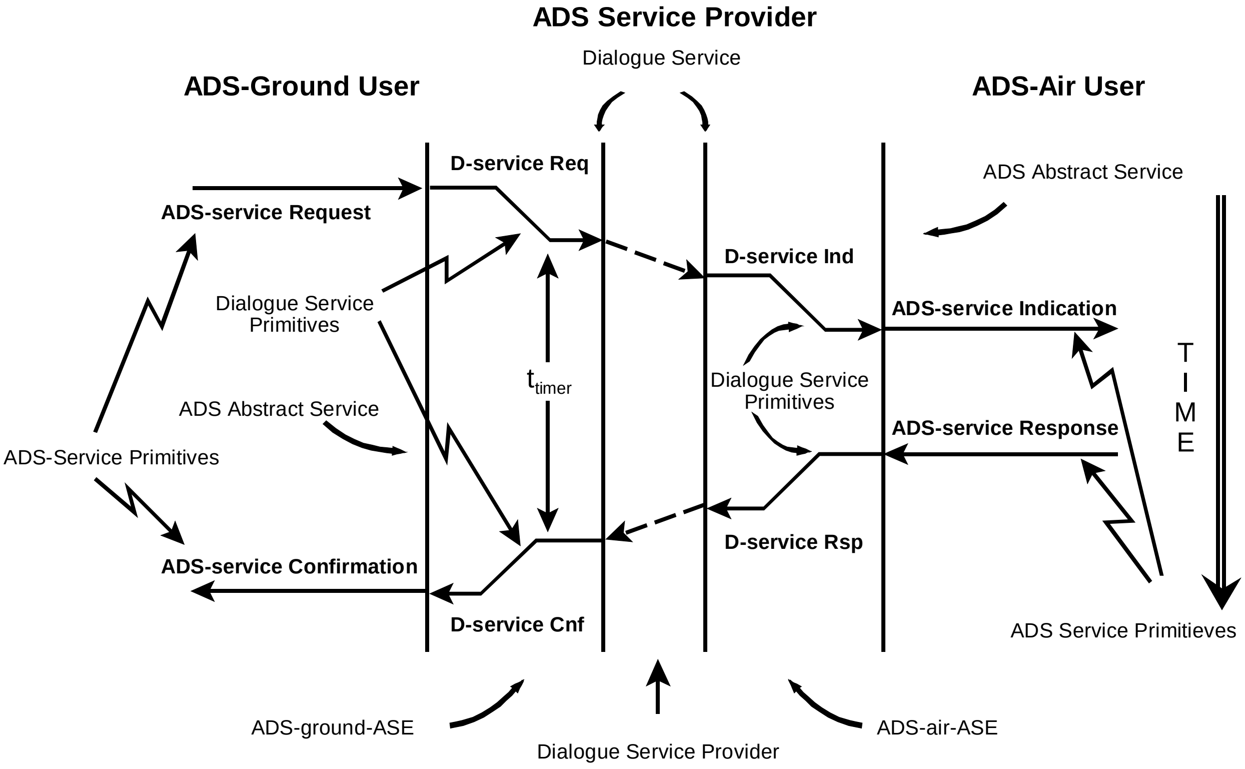


Figure 3.4-8. Message sequence diagram

3.4.4.1.9 Note that the time sequence diagrams in the SARPs representing abort situations can be overlaid on top of any of the other figures to represent an abort in action.

3.4.4.2 ***ADS SARPs 2.2.1.5.2 ADS Service Provider Timers***

3.4.4.2.1 This section lists the service provider timers that are defined in the protocol, and suggests values for them.

3.4.4.2.2 The purpose of the service provider timers in the ADS service provider is not operational. For operational reasons, there may be a requirement to have other timers that are shorter than the ones described here. The purpose of these service provider timers is only to ensure that the ASE protects itself when communicating with a system that has failed to respond.

3.4.4.2.3 Most, if not all, operational systems will require operational timers of much shorter duration than the values set for the service provider timers. The value of operational timers will also vary according to the operational scenario in which the aircraft is flying. The SARPs does not specify the value of the operational timers, nor does it insist on their implementation. For every operational system, values of operational timers will need to be calculated. Also, the action that must be taken if the operational timer expires must be specified. Examples of such actions are: an ADS-user-abort request is made and a connection is made with a different class of communications service; the connection could be left open while the controller contacts the aircraft via R/T; the colour of the aircraft on the user interface could change.

3.4.4.2.4 For example, suppose a ground system sends a demand contract to an aircraft. Suppose further that the aircraft system has not implemented the ADS protocol correctly, and it locks up without sending the result back to the ground system. The ground system will be in a state that is waiting for a result. The specification prevents the ground system from sending up another demand contract until the first has been dealt with. This combination of events would make the ground system lock up unless a service provider timer is used. When in this state, the t-DC-1 service provider timer will eventually reach its maximum value. The ground system can then abort the connection. Thus the ground system protects itself from getting locked up because of another system's failure.

3.4.4.2.5 Note that the values set in these timers have been calculated thus: It has been assumed that the slowest possible network is in operation and that all systems in the path have their maximum delay. Should a reply to a request take longer than this, then it is assumed that something must be failing somewhere.

3.4.4.2.6 The assignment of values for timers must be optimised based on operational testing of the application. In such testing, incompatible timer values and optimum combinations can be identified. Implementations of ADS protocol are required to support configurable values for all timers and protocol parameters, rather than having fixed values. This allows modification as operational experience is gained.

3.4.4.3 *SARPs Section 2.2.1.5.3 ADS-ASE Protocol Description*

3.4.4.3.1 The ADS service provider is described in the SARPs as finite state machines or protocol machines (PM). The protocol machine for a particular service starts in an initial state (IDLE). Events, which are service primitives received from the ADS-users above or the Dialogue Service provider below, as they occur, trigger activity on the part of the PM. As part of this activity, actions may be required (service primitives issued to the ADS-users and/or the underlying Dialogue Service provider).

3.4.4.3.2 The protocol description explains the rules by which the ASEs work. There is a detailed specification of actions taken by the ASEs when triggered by certain events:

- the arrival of a PDU through the dialogue service;
- the invocation of one of the service primitives by the user;
- the expiration of one of the internal technical timers;
- an unrecoverable system error.

3.4.4.3.3 In order to resolve a somewhat complex specification, both the ground and the air ASEs have been broken up into a number of internal modules. The behaviour of each is specified separately. In addition to the four triggers specified above, individual modules may also be triggered into action by the other modules within the same ASE.

3.4.4.3.4 Both air and ground ASEs have modules that mirror each other. However, apart from the AB module, the actions of the air and ground mirrored modules are different but complementary. The modules are:

- HI module — the main job of the HI module is to select which module must handle the primitives that are invoked by the user.
- LI module — the main jobs of the LI module are a) to select which module must handle the PDU passed to it from the dialogue service, and b) to manage the dialogue, selecting which dialogue service must be used at any given time.
- DC module — the job of the DC module is to manage demand contracts.
- EC module — the job of the EC module is to manage event contracts.
- PC module — the job of the PC module is to manage periodic contracts.
- AB module — the job of the AB module is to handle abort situations.

3.4.4.3.5 There is a requirement that the ASE does not accept the invocation of primitives when no actions are described for that primitive in that state (2.2.1.5.3.2). Some explanation of this statement is needed:

3.4.4.3.6 Each module has several different states. When a module is in a particular state, only some primitives are permitted to be invoked. For example, if an ADS-periodic-contract request is invoked, it is not permitted to invoke a second ADS-periodic-contract request until a reply has been received for the first one. There is no statement in the description of the protocol that explains what the ground ASE must do if it receives a second ADS-periodic-contract request before the reply to the first one has been received. The SARPs therefore requires (by statement 2.2.1.5.3.2), that the ground user must not invoke an ADS-periodic-contract request during this period.

3.4.4.3.7 Thus only actions which are permitted are described. If an action is not described, then it is not permitted.

3.4.4.4 *SARPs Section 2.2.1.5.5 ADS-ASE State Tables*

3.4.4.4.1 State tables are provided. These are an exact reflection of the ADS protocol description, in a condensed form. The state tables are only presented for guidance, since the textual protocol description always takes precedence.

3.4.5 SARPs Section 2.2.1.6 Communication Requirements

3.4.5.1 *SARPs Section 2.2.1.6.1 Encoding Rules*

3.4.5.1.1 This section states that PER (Packed encoding rules) must be used to encode the PDUs. PER is an ISO standard and is particularly efficient at encoding data. Implementors may use ASN.1 compilers to generate code that creates PER automatically.

3.4.5.2 *ADS SARPs 2.2.1.6.2 Dialogue Service Requirements*

3.4.5.2.1 The dialogue service requires a number of parameters to operate. SARPs sections 2.2.1.6 and 2.2.2.6 define those parameters that are not defined elsewhere.

3.4.5.2.2 In addition to the ClassOfCommunication parameter provided by the contract initiator, other Quality of Service (QOS) parameters are attached by the ASE to the dialogue supporting the communication. Values for the Application Priority and the Residual Error Rate are constant for the ADS application and therefore are not requested to the users.

3.4.5.2.3 The Application Priority is set by default to the abstract value “High Priority Flight Safety Messages”.

3.4.5.2.4 The Residual Error Rate is set by default to the abstract value “low”.

3.4.5.2.5 As the SARPs is only concerned with the use of ADS by Air Traffic Services, the only values for the ClassOfCommunication parameter that are possible for the user to choose are those that are used by ATS. The SARPs do not prevent the use of ADS by other (non-ATS) users, who would use non-ATS classes of communication service. However, since the SARPs are designed to have an ATS audience, non-ATS use of the SARPs is not discussed. The SARPs do require that an aircraft must always allow at least four connections from ATS ground systems. This does not prevent an ADS equipped aircraft having additional non-ATS systems connecting to it, provided this does not stop at least four ATS systems connecting.

3.4.5.2.6 The underlying layers provide a checksum and realise the requirement for message integrity. It is not necessary, therefore, to provide application level integrity by means such as redundancy checks and confirmation of services.

3.4.6 SARPs Section 2.2.1.7 ADS User Requirements

3.4.6.1 The requirements set out in this section are those defined in [4]. It explains how contracts are realised. For further explanation — see this manual.

3.4.6.1.1 Purpose of Section 2.2.1.7 and 2.2.2.7

3.4.6.1.2 The SARPs makes a clear distinction between what assures interoperability between ADS air and ground components, and that which is operationally acceptable. Sections 2.2.1.3 to

2.2.1.6 and 2.2.2.3 to 2.2.2.6 guarantee that different implementations will interoperate, but they do not guarantee that the service offered is operationally acceptable. For example, they ensure the correct exchange of a periodic contract, its acknowledgement, and subsequent reporting. They do not ensure that the information that is provided in the ADS reports is what is agreed in the contract, or that it is provided at the correct rate.

3.4.6.1.3 Sections 2.2.1.7 and 2.2.2.7 ensure that the ADS contracts behave in a way that is operationally acceptable. 2.2.1.7, for example, ensures that the information provided in a periodic contract is the information that is required by the contract and that the reporting rate is that required by the contract.

3.4.6.1.4 To clarify this further, 2.2.1.7 and 2.2.2.7 ensure a degree of operational acceptability. For example, although it ensures that ADS reports are delivered on time with the correct information, there is no assurance that the information is that which is required by the controller. The information required by the controller will be dependant upon the use that is being made of ADS at that particular time.

3.4.6.1.5 Other aspects of the ADS application that are not defined in 2.2.1.7 or 2.2.2.7 are the sources, required accuracy and latency of the information provided in an ADS report. For example, when a level is provided as part of the position parameter, there is no requirement stated that requires the information to be obtained from a certain type of equipment, whether it is a single value taken from the last reading from the equipment or an average of several readings taken in the last few seconds, and there is no requirement stating that the most recent equipment reading must have been taken within a particular number of seconds. These decisions are currently left up to the equipment manufacturer. It is anticipated that requirements will emerge in the future.

3.4.6.2 *Demand Contracts*

3.4.6.2.1 A demand contract allows the ground system to ask for a single ADS report from an aircraft. The operation of demand contracts is described in 1.5.2.

3.4.6.2.2 Note that, unlike event and periodic contracts, demand contracts do not permit the aircraft to return a positive acknowledgement, followed by an ADS report some time later. The reason for this is as follows: It is anticipated that normal operation will use event and/or periodic contracts. The use for demand contracts is, therefore, expected to be for cases when the controller needs an immediate response to a query outside the normal surveillance operation. The requirement is that the information is required as soon as possible. The aircraft can use the positive acknowledgement as a form of delay, and this is therefore prohibited in a demand contract.

3.4.6.2.3 If the ADS-air-user can satisfy the demand contract in its entirety and within a short space of time (0.5 seconds is recommended), then it sends an ADS-report immediately.

3.4.6.2.4 If the ADS-air-user supply some of the information required in the demand contract, but not all of it, then it sends a noncompliance notification immediately, followed by an ADS report which supplies the information that it has available. This only applies if some of the optional information is unavailable.

3.4.6.2.5 If the ADS-air-user cannot supply some of the mandatory information for an ADS-report, or cannot supply it within a short space of time (0.5 seconds is recommended), then it will reply with a negative acknowledgement.

3.4.6.3 *Event Contracts*

3.4.6.3.1 An event contract allows the airborne system to send one or more ADS reports if and when a given event occurs. The operation of event contracts is described in 1.5.3.

3.4.6.3.2 There are a number of different ways that an event contract can generate ADS reports:

a) All event contract that report on a variation from the current parameter value require a baseline report (e.g. heading change event). The baseline report is generated immediately on acceptance of the contract. Further reports are generated when the event occurs.

b) Some event contracts provide a single report when the event occurs (e.g. change in the next waypoint).

c) Some event contracts provide reports at a rate of one every sixty seconds while the aircraft is outside the parameters of the event contract. The aircraft will stop reporting as soon as it comes back into the parameters again (e.g. level range event).

3.4.6.3.3 In an event contract, the ground system has no choice over the information that is provided in the ADS report. The choice of optional fields within the ADS report is fixed by the type of contract. Thus the term “optional fields” is, in the case of event contracts, not a true reflection of the SARPs.

3.4.6.3.4 If the ADS-air-user is able to meet the terms of the event contract (that is, it is able to detect the events requested in the contract, and can supply the information required in the ADS report), then one of the following will occur:

- If the type of contract requires a baseline report (e.g. for a heading change event), or if the event has already occurred (e.g. the aircraft is already flying outside the level range given in the contract), then the ADS-air-user may send an ADS report which contains an acknowledgement of the contract immediately. Alternatively, the ADS-air-user could send a positive acknowledgement followed by an ADS report.

- If the type of contract does not require a baseline report, and if the event has not already occurred, then the ADS-air-user sends a positive acknowledgement immediately.

3.4.6.3.5 If the ADS-air-user is not able to detect some of the events requested in the contract, but it can detect some others, then it sends a noncompliance notification.

3.4.6.3.6 If the ADS-air-user cannot detect any of the events requested in the contract, or it is not able to generate the information required for the ADS reports which are sent when the event occurs, then it sends a negative acknowledgement.

3.4.6.4 *Periodic Contracts*

3.4.6.4.1 A periodic contract allows the aircraft to send ADS reports at regular intervals. The operation of periodic contracts is described in 1.5.4.

3.4.6.4.2 The periodic contract will be suspended by the aircraft if it initiated an emergency contract. The periodic contract is not operational until the emergency contract is ended.

3.4.6.4.3 If the ADS-air-user is able to generate the mandatory and the optional information required by the contract at the rate requested, then one of the following occurs:

- If the ADS-air-user is able to generate the first report required by the contract within a short space of time (0.5 seconds is recommended), then it does so.

- If it is not able to generate the first report required by the contract within a short space of time (0.5 seconds is recommended), then it sends a positive acknowledgement, followed by the first report at the time it is able to do so.

3.4.6.4.4 If the ADS-air-user is able to generate all the mandatory information in an ADS report, but is not able to generate some or all of the optional information that is requested, then it sends back a noncompliance notification followed by an ADS report.

3.4.6.4.5 If the ADS-air-user is able to generate all the mandatory information in an ADS report, but is not able to generate ADS reports at the rate that is requested, then it sends back a noncompliance notification followed by ADS reports at the default rate of 60 seconds.

3.4.6.4.6 If the ADS-air-user is not able to generate all the mandatory information in an ADS report, then it sends back a negative acknowledgement.

3.4.6.5 *Emergency Contracts*

3.4.6.5.1 An emergency contract allows the aircraft to send ADS emergency reports to ground systems at regular intervals during an emergency situation. The operation of emergency contracts is described in 1.5.6.

3.4.6.5.2 An emergency contract is initiated and cancelled by the airborne system. During the emergency contract, the ground system may cancel the event and/or periodic contract that was in operation at the beginning. In this situation, only the emergency contract is in place. Under these circumstances, when the aircraft cancels the emergency contract, the ground system will subsequently close the connection to the aircraft.

3.4.6.5.3 The ground system has very little control over the emergency contract. It cannot change the information that is reported in the emergency report, and it cannot cancel the contract without cancelling all contracts. It can only modify the emergency reporting rate, or cancel all the contracts.

3.4.6.5.4 The establishment of an emergency contract is done by sending an ADS emergency report. It is not acknowledged.

3.4.6.6 *Maximum Number of Connections*

3.4.6.6.1 An implementation of an airborne system will have an in-built maximum number of external connections that it can support. The SARPs require the airborne system to support contracts with at least four ground systems. This will require it to have the ability to accept at least five connections in the short term. The additional (fifth) connection must be available to allow the ADS-air-user to inform any ground system that attempts to establish a further contract, that it cannot do so.

3.4.6.6.2 The operational requirement from which this is derived states that the airborne system must allow contracts from at least four ATC systems. An implementation must, therefore, be able to distinguish between ATC connections from any other type of connection. It can do this by examining the class of communications service parameter, or the ICAO facility designation parameter.

3.4.6.6.3 One method of implementation is to reserve four connection “slots” for exclusive ATC usage. Connection attempts from non-ATC users would either have to be refused, or use non-reserved “slots”.

3.4.6.6.4 Another method of implementation is to allow all connection attempts to establish a connection. When the maximum number of connections has been made, the latest one is examined to see if it is an ATC connection. If it is, then the others are examined to see if they are ATC connections. If there are less than four ATC connections, then at least one of the non-ATC connections is aborted to allow the ATC connection to be made.

3.4.6.7 *Loss of Information Source*

3.4.6.7.1 If the information requested in a contract is not available because the equipment on board is not capable of providing the information, then the aircraft will return a noncompliance notification at the start of the contract.

3.4.6.7.2 If the information requested in a contract is available normally, however, during the operation of the contract it becomes unavailable, e.g. the equipment fails, then one of two things can happen: If the information is part of the basic report (i.e. position, timestamp or FOM) then this is considered a very serious error, and the aircraft aborts the connection. If the information is one of the optional elements, then this is considered a minor error. In this case, the aircraft simply omits the information from the report. The ground user knows what the report is supposed to contain, and can therefore determine that the information has become unavailable. It is up to the ground implementation to continue receiving incomplete reports, to request a new contract or to cancel the contract altogether.

3.4.6.8 ***Report Forwarding***

3.4.6.8.1 ADS report forwarding allows a ground system to forward ADS reports to another ground system. The operation of report forwarding is described in 1.5.9.

3.4.6.8.2 There is no negotiation over which of the reports to forward. The receiving system has to accept whatever is sent to it. Thus it is required that an agreement exists between the two ground systems beforehand, or through some non-standardised means, regarding which reports to forward and when.

3.4.7 **SARPs Section 2.2.1.8 Subsetting Rules**

3.4.7.1 There is some functionality within the ADS SARPs that ground system implementors may choose not to incorporate. For example, if a particular implementation of a ground system is designed always to use event and demand contract, then there is no requirement for it to have the code for periodic contracts implemented. An aircraft, on the other hand, must have the code to allow recognition of periodic requests, since it may fly into an area that does use periodic contracts.

3.4.7.2 There are some combinations of functionality that allow interoperability, and some that do not. This section defines the combinations of functionality that allow interoperability. Note that the aircraft must have full technical functionality to be SARPs compliant. (This does not imply that the system is not SARPs compliant, if part of the functionality is temporally unavailable.)

- The combinations of functionality are defined in a set of tables.

- Version number — only one version is defined. This is a placeholder for when future versions are defined.

- Protocol Options — this defines a number of options for parts of the protocol that may be implemented. The options may be implemented together. Each has a name associated with it — the predicate.

- ADS-ground-ASE configurations — this defines seven combinations of protocol options, each of which yields a coherent protocol.

- ADS-air-ASE configurations — this defines a single combination of protocol options, which is the only combination that yields a coherent protocol.

- Supported ADS service primitives — this defines the conditions under which the service primitives are applicable.

- Supported ADS APDUs — this defines the conditions under which the PDUs are applicable.

3.4.7.2.1 The subsetting rules define those subsets that are technically possible. The SARPs do not address the operational acceptability of these subsets. There is one exception to this: It is technically possible for an aircraft system to be implemented without emergency contracts. However, from an operational point of view this is unacceptable, and so has not been included as a valid subset.

3.4.7.3 The subsetting rules define subsets of the protocol. It is also possible to define subsets of user functionality beyond the rules defined in the SARPs. An aircraft may not be equipped to provide all the information required by a contract, and it may reply with a noncompliance notification or negative acknowledgement. For example, if an aircraft is not equipped to supply weather information, when a contract requests weather information, it may reply with a noncompliance notification. Similarly, if the aircraft is unable automatically to detect a particular event type, it may reply to a request for the event type with a negative acknowledgement. Under all circumstances, however, the aircraft must be able to recognise and process these contract requests, even if it cannot meet the terms of the contract. This type of subsetting is not covered in the SARPs Aircraft Functionality Options.

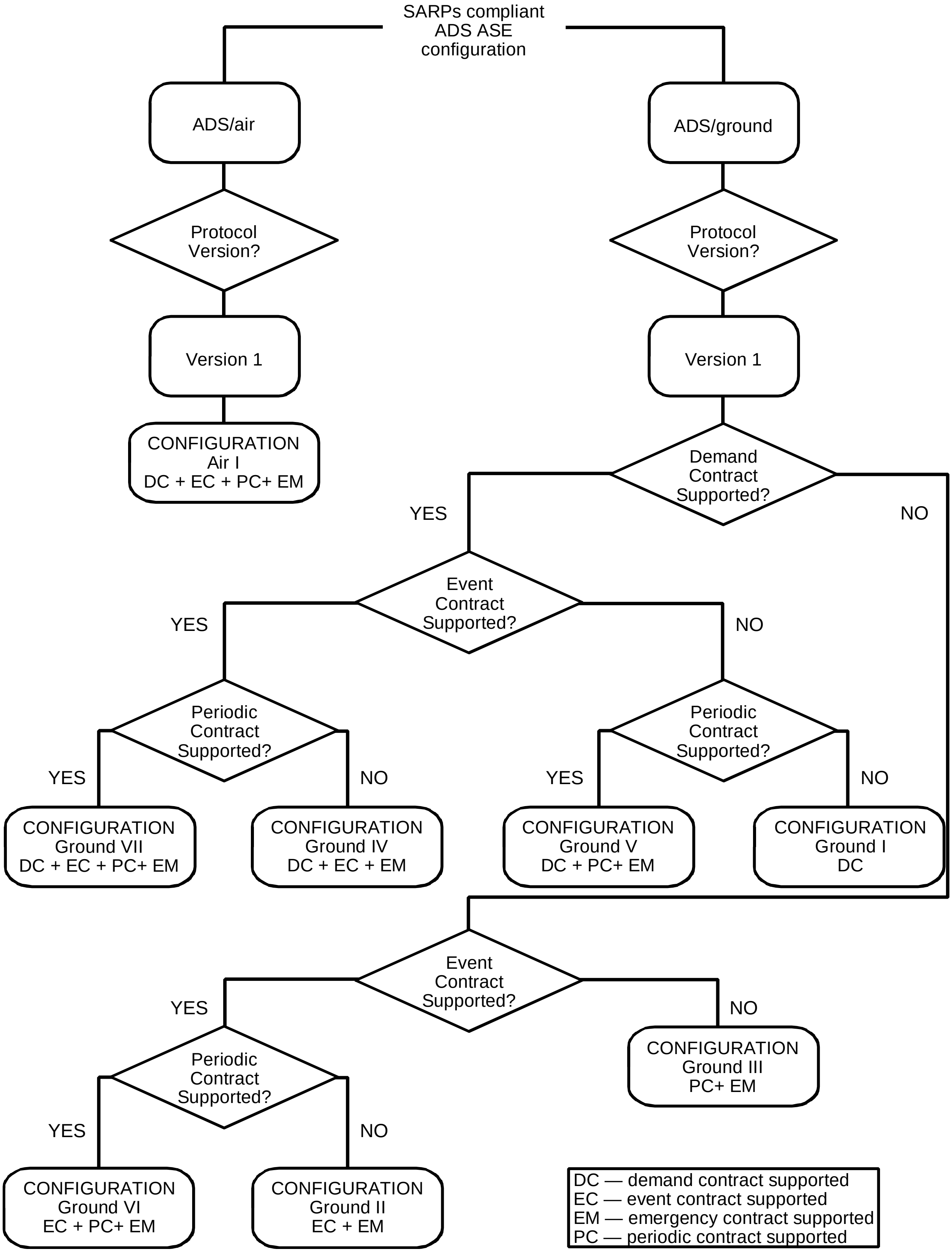
3.4.7.3.1 In the aircraft, all the functionality is mandatory, that is, there are no valid subsets other than implementing everything. This is because the aircraft must be able to support all contracts that the ground system requests of it.

3.4.7.3.2 It is theoretically possible for an aircraft system not to implement emergency contracts. However, this is considered operationally unacceptable, and is therefore not a valid option in the SARPs.

3.4.7.3.3 In the aircraft, it is possible to support part of the functionality of a contract — in that the equipment on board may not be able to handle a particular event or supply a particular piece of information. This optional functionality is handled by the use of the noncompliance notification. Such optional functionality is not covered in this section.

3.4.7.4 *Ground System Functionality Options*

3.4.7.4.1 The ground system does not have to implement the complete functionality described in the SARPs. A set of valid subsets is defined. These subsets have been chosen on the grounds of technical feasibility rather than a set of operational requirements.


SARPs compliant ADS ASE configuration
ADS/air
Protocol Version?
Version 1
CONFIGURATION Air I DC + EC + PC+ EM
ADS/ground
Protocol Version?
Version 1
Demand Contract Supported?
YES
NO
Event Contract Supported?
YES
NO
Periodic Contract Supported?
YES
NO
CONFIGURATION Ground VII DC + EC + PC+ EM
CONFIGURATION Ground IV DC + EC + EM
Periodic Contract Supported?
YES
NO
CONFIGURATION Ground V DC + PC+ EM
CONFIGURATION Ground I DC
Event Contract Supported?
YES
NO
CONFIGURATION Ground III PC+ EM
Periodic Contract Supported?
YES
NO
CONFIGURATION Ground VI EC + PC+ EM
CONFIGURATION Ground II EC + EM
DC — demand contract supported
EC — event contract supported
EM — emergency contract supported
PC — periodic contract supported

Examination of the options and comparison with the operational requirements of the system being built is therefore recommended.

3.4.7.4.2 The simplest option is that of only implementing the demand contract. This might be used in a scenario where ADS is not used for surveillance, but only used to check the extended projected profile against the filed flight plan for example.

3.4.7.4.3 All other options require the implementation of the emergency contract, since, if either the event or periodic contracts are implemented, then there is a possibility that the aircraft will initiate an emergency contract. The other functionality can include one of the following three combinations:

- event contract
- periodic contract
- both event and periodic contract

3.4.7.4.4 Each of these three options can be implemented either with demand contracts or without. Thus there are a total of seven options for ground systems.

3.4.7.4.5 Of course, the ground system may offer less than the full functionality of a given contract type. For example, a ground system periodic contract need never implement the request for intermediate-intent, and never implement the functionality of receiving an intermediate-intent.

3.4.7.5 *Report Forwarding Options*

3.4.7.5.1 When implementing ADS report forwarding, there are only two options — either implement an initiating system or a receiving system. A system that implements both initiating and receiving functions must implement them as two separate applications.

3.5 Dimensions

3.5.1 PDU Size

3.5.1.1 *Theoretical Limits*

3.5.1.1.1 Theoretically, the aircraft could be requested to provide a report with all possible types of information, including an extended projected profile with the maximum number of waypoints. The largest PDU that the aircraft will be required to produce will thus depend on the maximum number of waypoints that the flight management system can generate. Where the flight management system is capable of generating the maximum number of way points, then the largest APDU that the aircraft could generate is 1399 octets. The ground system must not request a report that is bigger than the maximum size that it can cope with.

3.5.1.1.2 The maximum APDU size for the ADS Report Forwarding application is only slightly bigger, and also fits into 1399 octets.

3.5.1.1.3 A major limiting factor with PDUs of this size is likely to be the bandwidth of the air-ground subnetwork. Current technology only provides limited bandwidth across the air-ground subnetwork, and so large amounts of data transmitted could flood it. The size of contract requests and other ground initiated PDUs are very small by comparison, and all airborne systems must be built with the capability to receive all requests from the ground.

3.5.1.2 A second limiting factor is likely to be the memory capacity of the airborne system. Many implementations will be built into existing equipment which have limited memory available for storage of the data structures used to build and examine PDUs. Such implementations may have to recognise when they cannot cope with a contract because of memory limitations, and respond with a noncompliance notification or negative acknowledgement Error Handling.

3.5.1.2.1 Should either airborne or ground system receive a PDU that is too large for it to manage under its current circumstances (e.g. lack of memory due to other applications), it will be unable to decode it. The system must abort the connection under these circumstances.

3.5.1.2.2 Table 3.5-1 gives typical sizes for the APDUs used in ADS.

Table 3.5-1. Typical sizes of APDUs

Scope	*APDU*	*Typical Size*	*Comments*
Uplink	aDS-cancel-all-contracts-PDU	1 octet	
	aDS-cancel-contract-PDU	1 octet	
	aDS-cancel-emergency-acknowledgement-PDU	1 octet	
	aDS-demand-contract-PDU	2 octets	This can be up to 4 octets if short term intent and/or extended projected profile is chosen.
	aDS-event-contract-PDU	5 octets	This can be up to 16 octets if all options are chosen; 5 octets covers 2 events only.
	aDS-modify-emergency-contract-PDU	2 octets	
	aDS-periodic-contract-PDU	5 octets	A contract for the basic report only is 3 octets; the addition of other optional blocks of information in the contract increases the APDU size to up to 13 octets.
	aDS-provider-abort-PDU	1 octet	
Downlink	aDS-cancel-emergency-PDU	1 octet	
	aDS-demand-report-PDU	15 octets	15 octets gives the basic report only (i.e. no optional information). With all the optional information extended to the maximum size, this can reach 1399 octets, although most of this is the full extended projected profile. Without this, the maximum size is 102 octets.
	aDS-emergency-report-PDU	15 octets	This can go up to 22 octets when aircraft address and ground vector information is included every fifth report.
	aDS-event-report-PDU	20 octets	20 octets covers the basic report together with a ground vector. The ASN.1 will allow this to reach a size of over 1399 octets. In practice, restrictions in the user part of the SARPs (2.2.1.7) prevent it being as big as this.
	aDS-negative-acknowledgement-PDU	2 octets	Normally this will be 2 octets — but it could be more if it is reporting that the maximum capacity is exceeded — in this case it will be at least 18 octets, possibly more can be expected.
	aDS-noncompliance-notification-PDU	20 octets	This can be up to 71 octets if there are a lot of noncompliances. 20 octets gives a single noncompliance.

Scope	*APDU*	*Typical Size*	*Comments*
	aDS-periodic-report-PDU	15 octets	15 octets gives the basic report only (i.e. no optional information). With all the optional information extended to the maximum size, this can reach 1399 octets, although most of this is the full extended projected profile. Without this, the maximum size is 102 octets.
	aDS-positive-acknowledgement-PDU	2 octets	
	aDS-provider-abort-PDU	2 octets	
Ground	aDS-forwarded-report-PDU	17 octets	17 octets gives the basic report only (i.e. no optional information). With all the optional information extended to the maximum size, this can reach 1402 octets, although most of this is the full extended projected profile.
	aDS-provider-abort-PDU	1 octet	

3.5.2 Rate of Message Transmission

3.5.2.1 *Limits*

3.5.2.1.1 The required maximum transmission rate for periodic and emergency contracts is one message per second. Thus, in theory, an aircraft must be able to handle transmission of the maximum size report at a rate of one per second to four different ATS ground systems. There are a number of factors that limit the rate of transmission, which include:

- the speed at which raw data can be gathered from the avionics;
- the processing power of the aircraft ADS system;
- the processing power of the airborne router;
- the bandwidth available on the air-ground subnetwork.

3.5.2.1.2 In practice, therefore, it is unlikely that this theoretical limit can be reached. It is for this reason that the aircraft is able to report to the ground that it is not able to meet the requested rate. In order for the airborne system to be able to calculate if it can meet the required reporting rate, it will have to have some information about the four factors listed above. Note that the fourth factor — the bandwidth available on the air-ground link — is dependent upon the subnetwork in operation at the time; this factor is therefore dynamic, whereas the first three are static. Some means of communicating the bandwidth available on the air-ground link to the ADS processor will have to be in place. The ground system controls the rate of report delivery on the periodic and emergency contracts. It must therefore request a rate that it can handle.

3.5.2.2 *Error Handling*

3.5.2.2.1 Should the aircraft be requested to deliver a rate that it determines is not possible, it will reply with a noncompliance notification and default to a 60 second rate. A condition may occur where a periodic contract has been accepted, and it is subsequently found that the reporting rate cannot be met. There is no way for the airborne system to negotiate a slower reporting rate once it has accepted it. There are two options available to the implementors:

- the aircraft could abort the connection;
- the aircraft could deliver at a slower rate.

3.5.2.2.2 If the bandwidth on the air-ground link is the problem, there may be congestion in the router, in which case it may be preferable to abort the connection. If the problem is obtaining raw data from the avionics, then it may be preferable to deliver at the slower speed, and allow the ground system to abort or change the rate as it desires. For the ground system, if it receives ADS reports at a slower rate than requested, the implementation has three options:

- the ground system could abort the connection;
- the ground system could request a new contract at a slower rate;
- the ground system could do nothing, accepting the rate as supplied.

3.5.3 Number of Connections

3.5.3.1 *Limits*

3.5.3.1.1 The SARPs requires that an aircraft is capable of managing at least four connections at the same time. That is, four ground systems may attempt to establish contracts, and the period of those contracts overlaps. Moreover, each of the ground systems may set up a periodic contract, an event contract, and a demand contract. Thus the aircraft must be able to handle all twelve contracts simultaneously. Although the aircraft must be able have contracts with at least four ground systems, there are no requirements that prevent an implementation managing more than four.

3.5.3.1.2 The operational requirement is for four ATSUs to have contracts. There may be other organisations, for example, the airline, which may also want to set up ADS contracts. In these circumstances, an implementation could be designed to manage more than four connections. In order to be compliant with the SARPs, the airborne system must reject a non-ATC system that is “using up” one of the four connections reserved for ATC systems. The only way the airborne system has of doing this is through examination of the ICAO facility designator. The airborne system must trust the ground based systems to set this up correctly.

3.5.3.1.3 There is no equivalent requirement for a ground system. In theory, a ground system implementation could manage only one connection. It seems unlikely that such an implementation would be useful. Careful consideration ought to be give to the requirements of the ground system before any limits are set.

3.5.3.1.4 In the air system, an attempt may be made to exceed the maximum number of connections set by the implementation (which must be four or more). In this case there is a requirement that the aircraft rejects the attempt to establish a contract while returning a list of the ground systems that already have connections. This will allow a controller to examine the list of connected systems, determine if there are any that are inappropriate, and contact them (for example, by telephone, or some automatic means) asking them to release their connection. The ground system initiates all ADS connections, and there is, therefore, no similar requirement of them. Of course, the ground system must have some local means of preventing more connections to be set up than it is designed to manage.

3.5.3.1.5 Ground forwarding can also be used to distribute the information to more ground systems.

3.5.3.1.6 For further guidance on sizes, see the *Manual of Air Traffic Services Data Link Applications* (Doc 9694) Part III, Chapter 3, Appendix A, Table 3A-1.

3.6 **ASN.1 Index**

3.6.1 **ASN.1 Section Index**

3.6.1.1 The ASN.1 discussed here is in two parts. The first part lies in section 2.2.1.4.2 of the ADS SARPs, and holds the main body of the ASN.1. In addition, there is a second section of ASN.1 in the ADS Report Forwarding Applications SARPs in 2.2.2.4.2.

3.6.1.2 Within the main body of the ASN.1, there are six sections divided up by comments crossing the page. Together with the ADS Report Forwarding ASN.1, there are seven sections. These have been given a section number S1 — S7, and are listed in the table below. These section numbers are not in the SARPs themselves, but have been included here to make the reading of the table in the following section easier. They can be used to determine the location of the ASN.1 type definition within the SARPs.

Table 3.6-1. ASN.1 Sections

Section reference	ASN.1 Section
S1	Aircraft generated and Ground generated message choice
S2	Ground generated and aircraft generated message components — protocol data units
S3	Reports and their components
S4	Components of contracts
S4	Miscellaneous components
S6	Common components
S7	ADS Report Forwarding Application

3.6.2 **ASN.1 Type Index**

3.6.2.1 The following table lists each ASN.1 type defined in the ADS SARPs in alphabetical order[1]. In order to enable the definition to be found more easily, a cross reference to the ASN.1 section is given.

3.6.2.2 The third column lists those ASN.1 types that are used in the definition of the ASN.1 type in the first column, and the fourth column lists those ASN.1 types that use it. The third and fourth columns are therefore inverse references.

3.6.2.3 For primitive types, the last two columns indicate the range and resolution of the type.

[1] In retrospect, it would have been better to present the ASN.1 itself in alphabetical order in the SARPs. However, it is not possible to make such dramatic changes once the document is under change control. Please accept the editor's apologies.

Table 3.6-2. ASN.1 Type Index

Type	ASN.1 Section	Types used by this type	Types using this type	Range	Resolution
AbortReason	S2	-	ADSAircraftPDUs ADSGroundPDUs ADSRFPDUs	-	-
ADSAircraftPDUs	S1	ADSDemandReport ADSEmergency ADSEventReport NegativeAcknowledgement ADSPeriodicReport PositiveAcknowledgement AbortReason	-	-	-
ADSDemandReport	S2	ADSReport	ADSAircraftPDUs	-	-
ADSEmergency	S2	ADSEmergencyReport	ADSAircraftPDUs	-	-
ADSEmergencyReport	S3	Position DataTimeGroup FigureOfMerit AircraftAddress GroundVector	ADSEmergency ForwardedReport	-	-
ADSEventReport	S2	EventTypeReported ADSReport	ADSAircraftPDUs	-	-
ADSForwardedReport	S7	AircraftAddress ForwardedReport	ADSRFPDUs	-	-
ADSGroundPDUs	S1	CancelContract DemandContract EventContract ModifyEmergency PeriodicContract AbortReason	-	-	-
ADSPeriodicReport	S2	ADSReport	ADSAircraftPDUs	-	-

Type	ASN.1 Section	Types used by this type	Types using this type	Range	Resolution
ADSReport	S3	Position DateTimeGroup FigureOfMerit AircraftAddress ProjectedProfile GroundVector AirVector Weather ShortTermIntent ExtendedProjectedProfile	ADSDemandReport ADSEventReport ADSPeriodicReport ForwardedReport	-	-
ADSRFPDUs	S7	ADSForwardedReport AbortReason	-	-	-
AircraftAddress	S3	-	ADSEmergencyReport ADSForwardedReport ADSReport	24 bits exactly	1 bit
AirSpeed	S3	Mach Ias	AirVector	-	-
AirSpeedChange	S4	-	EventContract	For Mach 0.005 - 1.275 For IAS 1 - nnnn	For Mach 0.005 Mach For IAS 1 knot
AirVector	S3	DegreesDirection AirSpeed VerticalRateChange	ADSReport	-	-
CancelContract	S2	-	ADSGroundPDUs	-	-
Date	S6	Year Month Day	DateTimeGroup	-	-
DateTimeGroup	S6	Date Time	ADSEmergencyReport ADSReport	-	-
Day	S6	-	Date	1 to 31	1 day
DegreesDirection	S4	-	AirVector EventContract GroundVector IntermediateIntent	0.1 to 360 degrees	0.1 degrees
DemandContract	S2	ProjectionTime ExtendedProjectedProfileRequest	ADSGroundPDUs	-	-

Type	ASN.1 Section	Types used by this type	Types using this type	Range	Resolution
Eta	S6	Time	ExtendedProjectedProfile ProjectedProfile	-	-
EventContract	S2	LateralChange VerticalRateChange LevelRange AirSpeedChange GroundSpeedChange DegreesDirection ExtendedProjectedProfileRequest LevelChange	ADSGroundPDUs	-	-
EventTypeContracted	S5	-	NoncomplianceNotification	-	-
EventTypeReported	S5	-	ADSEventReport ForwardedReport	-	-
ExtendedProjectedProfile	S3	Position Eta	ADSReport	-	-
ExtendedProjectedProfileModulus	S4	Modulus ExtendedProjectedProfileRequest	PeriodicContract	-	-
ExtendedProjectedProfileRequest	S4	-	DemandContract EventContract ExtendedProjectedProfileModulus	Time interval 15 minutes to 20 hours Number of waypoints 1 to 128	Time interval 15 minutes Number of waypoints 1
FigureOfMerit	S3	PositionAccuracy	ADSEmergencyReport ADSReport	-	-
ForwardedReport	S7	ADSReport EventTypeReported ADSEmergencyReport	ADSForwardedReport	-	-
GroundSpeedChange	S4	-	EventContract	0 to 300 knots	1 knot
GroundSystemsUsingService	S5	-	Reason	4 to 8 characters	1 character
GroundVector	S3	DegreesDirection VerticalRateChange	ADSEmergencyReport ADSReport	Ground-speed -50 to 2200 knots	Ground-speed 1 knot
Ias	S4	-	AirSpeed	0 to 1100 knots	1 knot
IntermediateIntent	S3	DegreesDirection Level ProjectionTime	ShortTermIntent	Distance 1 to 8000 nautical miles	Distance 1 nautical mile

Type	ASN.1 Section	Types used by this type	Types using this type	Range	Resolution
LateralChange	S4	-	EventContract	0 to 200 nautical miles	0.1 nautical miles
Latitude	S6	Sign	Position	degrees 0 to 90 Minutes 0 to 59 Tenth seconds 0 to 59.9 seconds	degrees 1 Minutes 1 Tenth seconds 0.1 seconds
Level	S6	-	IntermediateIntent LevelRange Position	-750 to 100000 feet	10 feet
LevelChange	S4	-	EventContract	10 to 5000 feet	10 feet
LevelRange	S4	Level	EventContract		
Longitude	S6	Sign	Position	degrees 0 to 180 Minutes 0 to 59 Tenth seconds 0 to 59.9 seconds	degrees 1 Minutes 1 Tenth seconds 0.1 seconds
Mach	S4	-	AirSpeed	0.5 to 4 mach	0.001 mach
ModifyEmergency	S2	ReportingInterval	ADSGroundPDUs	-	-
Modulus	S5	-	ExtendedProjectedProfileModulus PeriodicContract ShortTermIntentModulus	1 to 255	1
Month	S6	-	Date	1 to 12	1 month
NegativeAcknowledgement	S2	RequestType Reason	ADSAircraftPDUs	-	-
NoncomplianceNotification	S2	ReportType EventTypeContracted ReportTypeAndPeriod	ADSAircraftPDUs	-	-
PeriodicContract	S2	ReportingInterval Modulus ShortTermIntentModulus ExtendedProjectedProfileModulus	ADSGroundPDUs	-	-
Position	S6	Latitude Longitude Level	ADSEmergencyReport ADSReport ExtendedProjectedProfile ProjectedProfile ShortTermIntent	-	-
PositionAccuracy	S3	-	FigureOfMerit	-	-

Type	ASN.1 Section	Types used by this type	Types using this type	Range	Resolution
PositiveAcknowledgement	S2	RequestType	ADSAircraftPDUs	-	-
ProjectedProfile	S3	Position Eta	ADSReport	-	-
ProjectionTime	S4	-	DemandContract IntermediateIntent ShortTermIntent ShortTermIntentModulus	1 minute to four hours	1 minute
Reason	S5	GroundSystemsUsingService	NegativeAcknowledgement	-	-
ReportingInterval	S4	-	ModifyEmergency PeriodicContract	Seconds scale 1 to 59 seconds Minutes scale 1 to 120 minutes	Seconds scale 1 second Minutes scale 1 minute
ReportType	S5	-	NoncomplianceNotification	-	-
ReportTypeAndPeriod	S5	-	NoncomplianceNotification	-	-
RequestType	S5	-	NegativeAcknowledgement PositiveAcknowledgement	-	-
ShortTermIntent	S3	Position ProjectionTime IntermediateIntent	ADSReport	-	-
ShortTermIntentModulus	S4	Modulus ProjectionTime	PeriodicContract	-	-
Sign	S6	-	Latitude Longitude	-	-
Time	S6	TimeHours TimeMinutes TimeSeconds	DateTimeGroup Eta	-	-
TimeHours	S6	-	Time	0 to 23 hours	1 hour
TimeMinutes	S6	-	Time	0 to 59 minutes	1 minute
TimeSeconds	S6	-	Time	0 to 59 seconds	1 second
VerticalRateChange	S4	-	AirVector EventContract GroundVector	-30000 to 30000 feet per minute	10 feet per minute

Type	ASN.1 Section	Types used by this type	Types using this type	Range	Resolution
Weather	S3	-	ADSReport	Wind speed 0 to 300 knots Wind direction 1 to 360 degrees Temperature -100 to 100 degrees Centigrade Turbulence 0 to 15 (units to be decided)	Wind speed 1 knot Wind direction 1 degree Temperature 0.25 degrees Centigrade Turbulence 1 unit
Year	S6	-	Date	1996 to 2095	1 year

3.6.3 **ASN.1 Glossary.**

ADS Event Report: ADS information consisting of a sequence of Event Type and ADS Report.

ADS Emergency Report: ADS information consisting of the following sequence:

a) Position,

b) Time Stamp,

c) FOM,

d) Aircraft Identification (optional), and

e) Ground Vector (optional).

ADS Report: ADS information consisting of the following sequence:

[1] Position,

[2] Time Stamp,

[3] FOM,

[4] Aircraft Identification (optional),

[5] Projected Profile (optional),

[6] Ground Vector (optional),

[7] Air Vector (optional),

[8] Meteorological Information (optional),

[9] Short Term Intent (optional), and

[10] Extended Projected Profile (optional).

Aircraft Identification: Field 7 of the ICAO flight plan.

Air Speed: Provides airspeed as a choice of the following: Mach, IAS, or Mach and IAS.

Air Speed Change: Provides the threshold of change for either Mach speed or indicated air speed that requires that an avionics generate an ADS report when the current aircraft speed differs more than the specified threshold from the air speed in the last ADS report.

Air Vector: Provides the air vector as a sequence of Heading, Air Speed, and Vertical Rate.

Cancel Contract: Allow the ground to cancel event and/or periodic contracts in effect.

Contract Type: Indicates which type of ADS contract is specified: demand, event, or periodic.

Demand Contract: Indicates that an avionics is to generate an ADS report containing the indicated data upon receipt of the contract. The data that can be indicated includes: aircraft identification, projected profile, ground vector, air vector, meteorological information, short term intent, and extended projected profile.

Distance: Distance.

ETA: Estimated time of arrival at a waypoint.

Event Contract: Indicates Event Types and the threshold for the specified event types.

Event Type: An indication of what type of ADS event is specified:

- Vertical rate change,
- Way-point change,
- Lateral deviation change,
- Level change,
- Level range deviation,
- Airspeed change,
- Ground speed change,
- Heading change,
- Extended projected profile change,
- FOM (Figure of Merit) field change, and
- Track angle change.

Extended Projected Profile: Provides a sequence (1-128) of waypoint position data and ETA at the specified waypoint.

Extended Projected Profile Change: Indicates that an ADS report be generated when there is a change in the extended projected profile.

Extended Projected Profile Modulus: Sequence of Modulus and Extended Projected Profile Request.

Extended Projected Profile Request: a choice indicating whether the extended projected profile information is to be provided on a time or waypoint interval, and the interval of the specified choice.

Following Way Point: Indicates the waypoint after the next way point.

FOM: Indicates the figure of merit of the current ADS data. The information consists of the Position Accuracy and indications 1) whether or not multiple navigational units are operating, and 2) whether or not ACAS is available.

FOM Field Change: Indicates that an ADS report be generated when any FOM field changes.

Ground Speed: Provides ground speed.

Ground Speed Change: Provides the threshold of change for ground speed that requires the avionics to generate an ADS report when the current aircraft ground speed has differed by more than the specified threshold from the last ADS report.

Ground Vector: Provides the ground vector of an aircraft provided as a sequence of Track, Ground Speed, and Vertical Rate.

Heading: Provides aircraft heading in degrees.

Heading Change: Provides the threshold of change for heading in degrees that requires the avionics to generate an ADS report when the current heading has differed by more than the specified threshold from the last ADS report.

IAS: Indicated air speed.

ICAO Facility Designator: The 8 letter code which uniquely defines an ICAO ATSU facility.

Intermediate Intent: Set of points between current position and the time indicated in the Short Term Intent. Consists of a sequence of the following: Distance, Track, Level and Projected Time.

Lateral Deviation Change: Provides the threshold of change for lateral value that requires the avionics to generate an ADS report when the current lateral deviation exceeds the specified threshold.

Latitude: Latitude in degrees, minutes, and seconds.

Longitude: Longitude in degrees, minutes, and seconds.

Level: Specifies level.

Level Ceiling: The level above which an Level Deviation Event is triggered. Provided as an Level.

Level Change: Provides the threshold of change for level that requires the avionics to generate an ADS report when the current level differs by more than the specified threshold from the level in the last ADS report.

Level Floor: The level below which an Level Deviation Event is triggered. Provided as an Level.

Level Range Change: Threshold of change permissible between levels in consecutive ADS reports.

Mach: Air speed given as a Mach number.

Mach and IAS: Air speed provided as both Mach and Indicated Air Speed.

Meteorological Information: A sequence of Wind Direction, Wind Speed, Temperature and Turbulence.

Modulus: Provides a multiplier on the basic ADS report interval.

Next Time: Time at next waypoint.

Next Way Point: specifies the next waypoint in the avionics.

Noncompliance Notification: used to indicate partial compliance to a contract.

Periodic Contract: Provides the requirements for the generation of ADS reports. The periodic contract provides the reporting interval, and the modulus for when and what optional data is included in an ADS periodic report.

Position: Provides aircraft position information using a sequence of Latitude, Longitude, and Level.

Position Accuracy: An indication of the navigational accuracy.

Projected Profile: A sequence of Next Way Point, Next Time, and Following Waypoint.

Projected Time: Predicted time at a particular point.

Reporting Interval: Provides the required ADS reporting interval.

Report Type: Indicates which type of ADS report is provided: demand, event or periodic.

Request Type: A choice indicating which type of ADS request is being uplinked. The choices are as indicated below:

- cancel event contract,
- cancel periodic contract,
- demand contract,
- event contract,
- cancel emergency,
- modify emergency,
- periodic contract, or
- cancel all contracts.

Short Term Intent: A sequence of Position, ETA, and Intermediate Intent (optional) data structures.

Temperature: Temperature in degrees Celsius.

Track: Provides track angle in degrees.

Track Angle Change: Provides the threshold of change for track angle in degrees which triggers avionics to generate an ADS report when the current track angle differs by more than the specified threshold from the track angle in the last ADS report.

Turbulence: Indicates severity of turbulence on a scale of 0-15.

Vertical Rate: Rate of climb/descent (climb positive, descent negative).

Vertical Rate Change: The threshold of change for vertical rate that requires the avionics to generate an ADS report when the current vertical rate differs by more than the specified threshold from the vertical rate in the last ADS report.

Way Point Change: Change in the next waypoint information.

Wind Direction: Wind direction in degrees.

Wind Speed: Wind speed.

3.7 Example Scenarios

3.7.1 Introduction

3.7.1.1 This section contains a set of example scenarios of use. The purpose of this section is to demonstrate some scenarios that are theoretically possible using ADS. It is not meant to indicate what is required from an operational point of view.

3.7.2 Demand Contract in Isolation

3.7.2.1 A ground system requests an aircraft to provide an extended projected profile for its complete flight in a demand contract, and the aircraft returns its extended projected profile. This scenario could be used by a ground system that wants to compare the aircraft's flight plan with its own record of the flight plan stored in the flight data processing system. This could be initiated automatically for every aircraft as it approaches the FIR boundary. The controller may be unaware of this activity taking place.

3.7.3 Periodic Contract in Isolation

3.7.3.1 A ground system requests an aircraft to provide a periodic contract with:

- a period of 20 seconds;
- a ground vector every 15th report (i.e. every 5 minutes);
- weather information every 36th report (i.e. every 12 minutes).

3.7.3.2 This scenario could be used by a ground system that wants to track an aircraft closely over non-radar airspace. The weather information could be used to provide early indication of poor conditions for another aircraft following the first.

3.7.4 Event Contract in Isolation

3.7.4.1 A ground system requests an aircraft to provide an event report under the following conditions:

- the aircraft goes above 37 300 feet or below 36 700 feet (level range);
- the aircraft deviates from its course laterally by more than half a nautical mile (lateral deviation change);
- the aircraft's navigational accuracy changes (FOM change).

3.7.4.2 The first two constraints conceptually box the aircraft in. No reports are generated provided that it does not deviate outside the level range, nor half a nautical mile laterally and the FOM does not change. This creates a three dimensional tube through which the aircraft can fly. Reports are generated if the be unable to determine if it is within these limits with the same degree of accuracy. In such a case, a report aircraft leaves this zone. Should some of the aircraft's navigational equipment fail, it may will be generated to inform the controller of the change in the ability of the aircraft to determine its position. The controller will thus be able to make allowances for aircraft deviations and/or degraded performance. Note that the event will also be triggered if the ACAS availability changes.

3.7.5 Periodic and Emergency Contract

3.7.5.1 A ground system has established a periodic contract with an aircraft reporting at 90-second intervals. During the operation of this contract, an emergency occurs on the aircraft, such as the failure of an engine. The avionics detects this failure and automatically signals the ADS system. The ADS system immediately suspends the periodic contract with the ground system. It then initiates an emergency contract, reporting at 45-second intervals. The ground system indicates to the controller that an emergency is in place (say by showing the aircraft in a different colour). This allows the controller to contact the aircraft through a voice link or CPDLC in order to determine what to do. If the emergency is not detectable by the avionics (e.g. a medical emergency of one of the passengers or crew), the pilot can initiate the emergency contract. If the emergency is rectified and the pilot cancels the emergency contract, the ADS system reverts to the periodic contract it had with the ground system beforehand.

3.8 Example Encoding

3.8.1 ADS-demand-contract PDU

3.8.1.1 The following is an example of the encoding of an ADS-demand-contract-PDU. The demand contract requests the following information:

- aircraft address
- air vector
- extended projected profile for the next 30 way points

Table 3.8-1. ADS-demand-contract-PDU encoding

Element	*Sub-element*	*Value*	*Encoding*	*Comments*
ADSGroundPDUs CHOICE		aDS-demand-contract-PDU [3]	0 000011	no extension used the CHOICE 3 is selected
SEQUENCE			0 1001001	no extension used selection of 1st, 4th and 7th elements
aircraft-address [0]		NULL		no encoding for NULL
project-profile[1]				

Element	*Sub-element*	*Value*	*Encoding*	*Comments*
ground-vector [2]				
air-vector [3]		NULL		no encoding for NULL
weather [4]				
short-term-intent [5]				
extended-projected-profile [6]				
	CHOICE	number-way-points [1]	1	the CHOICE 1 is selected
	INTEGER	30	0011101	the INTEGER 30 encoded as 30-1 (i.e.29), since 1 is the lower bound of the range

3.8.1.1.1 Thus the PDU is encoded as 00000110 10010011 00111010, or in hexadecimal: 06 93 3A.

4. *CPDLC APPLICATION*

4.1 Introduction

4.1.1 Purpose

4.1.1.1 In line with normal ICAO practice, this chapter of Part III was developed as part of a companion document to the ATN Controller Pilot Data Link Communication (CPDLC) Standards And Recommended Practices (SARPs). It may be read alongside the CPDLC SARPs, in order to provide a greater understanding of the specification itself, or it may be read instead of the CPDLC SARPs by readers that simply want to understand the purpose of the CPDLC Application rather than the detail of the specification.

4.1.1.2 This chapter also provides some historical information on the development of the CPDLC Application and explanations as to why the CPDLC Application is specified the way it is, including corresponding notes an recommendations, in the SARPs.

4.1.2 Scope

4.1.2.1 This chapter provides guidance material for those implementing the CPDLC Application.

4.1.2.2 This chapter does not define any mandatory or optional requirements for CPDLC, neither does it define any recommended practices. This chapter does not instruct users on how to use the CPDLC Application in a particular operational environment.

4.1.2.3 The CPDLC application SARPs deal exclusively with Air Traffic Services (ATS). Aeronautical Operational Control (AOC) may choose to use the CPDLC SARPs as a model for their own applications.

4.1.3 History

4.1.3.1 The CPDLC application allows a pilot and controller to exchange messages. In a fully operational ATS data link environment, CPDLC is expected to be used as the main means of pilot-controller dialogue with voice available for backup.

4.1.3.2 The ADS Panel has developed the operational requirements for the CPDLC application over a period of several years. These requirements are specified in the *Manual of Air Traffic Services Data Link Applications* (Doc 9694) [3]. In addition to the operational requirements specified for CPDLC, the ADSP has specified that the CPDLC application should conform to the ATN protocols for its data link operations.

4.1.3.3 In the initial implementation of CPDLC, CPDLC messages, their format and intent, were based on the relevant ICAO documentation, in particular Annexes 2 and 11 and the *Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services* (PANS-RAC, Doc 4444). The format and content of the messages was developed based on the voice message set documented in Doc 4444.

4.1.3.4 The initial development of the CPDLC SARPs was relatively straight-forward: the exchange of data-link messages between a pilot and controller. However, it was recognized that the use of data link is not as flexible as voice, and a set of rules has had to be developed indicating, for example, how a dialogue is opened and closed, how CPDLC dialogues transition between data authorities, how communication with downstream data authorities could take place, and how CPDLC messages could be exchanged ground-ground, to complement the basic air-ground nature of CPDLC.

4.1.3.5 Some of the CPDLC capabilities such as the use of downstream clearances and ground-ground CPDLC message forwarding were seen as having limited applicability. States would not be willing to incur costs of implementing the complete CPDLC if they only intended to use certain elements of the application. The SARPs therefore took account of the need to separate out the functionalities to enable partial implementation, while still retaining the interoperability required by the ICAO Standards. This led to the development of subsetting rules, and the identification of conformant configurations.

4.1.3.6 The ATNP worked very closely with the ADSP to ensure that the development of both the operational concepts and the technical means of achieving them are consistent. However, the ADSP generally looked at a longer time scale than the current ATNP initial implementation program, and this means that some elements of their work have not been incorporated into the present SARPs.

4.1.3.7 The ATNP identified a set of packages to accommodate the continuous specification of operational requirements by the ADSP. This chapter provides guidance material for Version 1 of the CPDLC application contained in the first set of ATN SARPs (known as CNS/ATM-1, Package).

4.1.4 **Structure**

4.1.4.1 Chapter 4.1 — Introduction — This chapter contains the reason for providing guidance material as well as the scope. In addition, it provides a brief overview of the CPDLC functionality, CPDLC's relationship with other SARPs, and identifies applicable reference documents.

4.1.4.2 Chapter 4.2 — Overall General Functionality — This chapter describes generic concepts that are used throughout the CPDLC SARPs and guidance material. This chapter also covers some implementation issues that are not addressed in the SARPs.

4.1.4.3 Chapter 4.3 — CPDLC Service Description — This chapter gives a functional breakdown of the various services that CPDLC provides. It describes a peer to peer interaction, including reasons for why particular information is used or not used, and what actions on the information are expected.

4.1.4.4 Chapter 4.4 — CPDLC SARPs Section Description — This chapter clarifies any functionality that was not addressed in Chapter 3 on a CPDLC SARPs section by section basis.

4.1.4.5 Chapter 4.5 — Dimensions — This chapter gives some sample encoding sizes for guidance on what capacities need to be allowed for in order to meet normal operational expectations.

4.1.4.6 Chapter 4.6 — Indexes/Tables — This chapter provides a CPDLC variables glossary, and hierarchical relationships between the CPDLC message elements and variables.

4.1.4.7 Chapter 4.7 — Example Scenarios — This chapter gives some examples as to what typical scenarios are expected in course of normal CPDLC operation.

4.1.4.8 Chapter 4.8 — Example Encoding — This chapter outlines some actual sample PER encoding of typical CPDLC messages.

4.1.5 **CPDLC Application Overview**

4.1.5.1 This chapter gives a high level overview of the CPDLC application, as an application whose primary function is to allow the data link exchange of messages between a controller and a pilot. It contains an outline description of the functions which the CPDLC application provides, namely:

a) Controller-Pilot Message Exchange Function — allows pilots and controller to communicate via data link,

b) Transfer of Data Authority Function — allows a transfer in data authority without loss of an available CPDLC data link connection,

c) Down Stream Clearance Function — allows receipt of clearances affecting an aircraft’s future flight plan from a data authority not having current data authority, and

d) Ground Forward Function — allows a ground system to forward CPDLC messages to another ground system.

4.1.5.2 ***Controller-Pilot Message Exchange Function***

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

a) general information exchange;

b) clearance

1) delivery,

2) request, and

3) response;

c) altitude/identity surveillance;

d) monitoring of current/planned position;

e) advisories

1) request and

2) delivery;

f) system management functions; and

g) emergency situations.

4.1.5.3 *Transfer of Data Authority Function*

4.1.5.3.1 The Transfer of Data Authority Function provides the capability for a smooth transition of the CPDLC function when an aircraft transfers from one data authority to the subsequent one. The data authority having the capability to have an active (exchange of CPDLC messages affecting the aircraft's current flight path) CPDLC dialogue with an aircraft is known as the Current Data Authority (CDA). A given aircraft may only have one CDA at a time. Only a CDA is allowed to designate another ground system as the Next Data Authority (NDA). An inactive (no exchange of CPDLC messages) CPDLC dialogue can be opened with or by the NDA at a time before it becomes the CDA. This capability is intended to prevent a loss of communication that would occur if a NDA were prevented from actually establishing a dialogue with an aircraft until it became the CDA. The designation of a NDA is accomplished using a CPDLC message. A CDA is not required to designate a NDA, in which case no NDA connection is permitted to be established. The CDA may also change the NDA, in which case any connection in place with the now previous NDA is terminated, and a connection is established with the newly designated NDA. The CDA may also cancel a NDA without designating another NDA, in which case any connection in place with the canceled NDA is terminated. An aircraft may have only one NDA connection at a time.

4.1.5.3.2 A transfer of control can be accomplished without a transfer of data authority taking place. This can occur when more than one control authority is defined within a data authority. In this case, the receiving control area is not designated as an NDA, but is in fact still the CDA. The ground CPDLC application has the capability to inform the airborne CPDLC application of the transfer of control using CPDLC messages. This operational situation is not addressed by the CPDLC SARPs, since although operationally significant, it has no technical communication requirements beyond those already outlined for CDA communications.

4.1.5.4 *Down Stream Clearance Function*

4.1.5.4.1 The Down Stream Clearance Function provides the capability for an aircraft to contact an air traffic service unit which is not the current data authority, but is expected to be in the future, for the purpose of receiving a DownStream Clearance (DSC). This data authority is designated the Downstream Data Authority (DDA). Downstream clearance information is exchanged using CPDLC message(s). Care must be taken that downstream clearance exchanges do not affect an aircraft's current flight trajectory. The DSC function could also be accomplished using ground-ground connectivity. In this configuration, the CDA obtains the required information from a DDA and then provides it to an aircraft using the CDA CPDLC connection. The DSC functionality has been included since it has been recognized that ground-ground connectivity is not global.

4.1.5.5 *Ground Forward Function*

4.1.5.6 The Ground Forward Function provides the capability for a ground system to forward a CPDLC message to another ground system. The ground forward 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. A ground forward CPDLC connection can be used to forward one CPDLC message and is then terminated.

4.1.5.7 *Mapping of CPDLC Functionality To SARPs CPDLC Services*

4.1.5.7.1 Unlike the other application SARPs (CM, ADS, and FIS), there is not a one-to-one mapping (except abort services) between the CPDLC functions (operational) described above and the CPDLC services (technical) defined in the SARPs. This is due to the fact that all of the above functions exchange a common set of CPDLC messages and have common technical capabilities. Thus the SARPs defines CPDLC services to allow the message exchange between air-ground or ground-ground. A mapping showing the CPDLC services used by a given CPDLC function is provided in Table 4.1-1 below.

Table 4.1-1. Mapping of CPDLC Services to CPDLC Functions

CPDLC Function	CPDLC SARPs Service
Controller Pilot Message Exchange	CPDLC-Start Service CPDLC-Message Service CPDLC-End Service CPDLC-User-Abort Service CPDLC-Provider-Abort Service
Transfer of Data Authority	CPDLC-Start Service CPDLC-Message Service CPDLC-End Service CPDLC-User-Abort Service CPDLC-Provider-Abort Service
DownStream Clearance Delivery	DSC-Start Service CPDLC-Message Service DSC-End Service CPDLC-User-Abort Service CPDLC-Provider-Abort Service
Ground Forward	CPDLC-Forward Service CPDLC-User-Abort Service CPDLC-Provider-Abort Service

4.1.6 Inter-relationships with Other SARPs

4.1.6.1 There is no interaction between the CPDLC SARPs and the other CNS/ATM-1 Application SARPs.

4.1.6.2 The CPDLC SARPs make use of the Upper Layer Application SARPs [2] to perform dialogue service functions required by the CPDLC application.

4.1.7 Structure of the SARPs

4.1.7.1 All the air-ground SARPs are produced to a standard format. This has greatly helped the maintenance of document stability, commonality and presentation. The CPDLC SARPs are no different in basic layout from all other air-ground applications SARPs.

4.1.7.2 The CPDLC SARPs constitute the third part of sub-volume 2.

4.1.7.3 CPDLC SARPs Section 2.3.1 — Introduction — gives a very brief, high level description of CPDLC, as an application enabling CPDLC services to be provided to a pilot and controller via the exchange of messages between aircraft avionics and ground CPDLC systems. Since this overview contains no information directly related to the stipulation of specific standards, it is almost entirely written as series of informative notes.

4.1.7.4 CPDLC SARPs Section 2.3.2 — General Requirements — contains information and high level requirements for error processing and CPDLC version number requirements.

4.1.7.5 CPDLC SARPs Section 2.3.3 — Abstract Service — defines the abstract service interface for the CPDLC Application. The CPDLC Application Service Element (CPDLC-ASE) abstract service is described from the viewpoint of the CPDLC-air-user, the CPDLC-ground-user, and the CPDLC-service-provider.

4.1.7.6 CPDLC SARPs Section 2.3.4 — Formal Definition of Messages — describes the contents of all permissible CPDLC messages through definition of the CPDLC ASN.1 abstract syntax. All possible combinations of message parameters and their range of values are detailed.

4.1.7.7 CPDLC SARPs Section 2.3.5 — Protocol Definition — splits up the specification of the CPDLC protocol into three parts: sequence diagrams for the services covered by the abstract service, protocol descriptions and error handling for the CPDLC-Air-ASE and CPDLC-Ground-ASE, and State Tables.

4.1.7.8 CPDLC SARPs Section 2.3.6 — Communication Requirements — specifies the use of Packed Encoding Rules (PER) to encode/decode the ASN.1 message structure and stipulates the Dialogue Service requirements, including Quality of Service (QOS).

4.1.7.9 CPDLC SARPs Section 2.3.7 — CPDLC User Requirements — describes the requirements imposed on the CPDLC-users concerning CPDLC messages and interfacing with the CPDLC-ASEs.

4.1.7.10 CPDLC SARPs Section 2.3.8 — Subsetting Rules — specifies conformance requirements which all implementations of the CPDLC protocol obey. The protocol options are tabulated, and indication is given as to whether mandatory, optional or conditional support is required to ensure conformance to the SARPs. These subsetting rules permit applications to be tailored to suit individual implementations, commensurate with the underlying task, while still maintaining an acceptable level of interoperability.

4.1.8 References

Controller Pilot Data Link Communication Application, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705) — Air-Ground Applications, Section 2.3

[1] Upper Layers Communications Service, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume IV of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[2] *Manual of Air Traffic Services Data Link Applications* (Doc 9694)

[3] Introduction and System Level Requirements, Annex 10 — *Aeronautical Telecommunications,* Volume III, Part I, Chapter 3 and Sub-Volume I of the *Manual of Technical Provisions for the ATN* (Doc 9705)

4.2 Overall General Functionality

4.2.1 General

4.2.1.1 CPDLC is defined as a single application handling: CPDLC air-ground exchanges, DSC air-ground exchanges, and CPDLC ground-ground exchanges.

4.2.2 Topology

4.2.2.1 *Air-Ground Links*

4.2.2.1.1 The primary connection that is made in CPDLC is between an aircraft and the ground system responsible for control of the aircraft. This ground system is designated the Current Data Authority (CDA). This connection can be initiated by either the air or the ground. An aircraft can have only one CDA connection at any one time. The CDA connection is the only CPDLC connection permitting the exchange of control information that can affect the aircraft's current flight path. With the exception of abort conditions, the ground system is always responsible for initiating connection closure. While the CDA connection is active either the pilot (airborne system) or controller (ground system) can initiate CPDLC message exchanges. This is core functionality of CPDLC and must be supported by all CPDLC ground and airborne systems.

4.2.2.1.2 The CPDLC application can be used when more than one ground control authority is contained within the same data authority and uses the same CPDLC air-ground link. In this case there is a transfer(s) of control authority without a corresponding transfer of data authority. Any transfer of control authority on the ground with this topology is governed by local procedures. The CPDLC SARPs do not address this topology, since technically this is all part of a given CDA link. However, the CPDLC application message set does contain a message element (UM163: [facilitydesignation]) to allow the ground system to indicate to the airborne system that the control authority has changed, although the CDA link remains unchanged. When this topology is in effect a NDA should not be assigned until the next control authority is not part of the CDA.

4.2.2.1.3 In addition to the CDA connection, a given aircraft can have a CPDLC connection with another ground system if instructed to do so by the CDA. This secondary connection is between the aircraft and the Next Data Authority (NDA) as designated by the CDA. This connection can be initiated by either the air or the ground. An aircraft can have only one NDA connection at any one time, and this can only be established under direction of the CDA. The NDA connection CANNOT be used to exchange any operational messages. The NDA connection is used to facilitate the transfer of data authority between ground systems without loss of CPDLC connectivity on the airborne side. With the exception of abort conditions, a NDA connection is not terminated by either the airborne or ground side, but rather it is converted to a CDA connection upon normal (non-abort) termination of the

CDA connection. (Note that instructions from the CDA canceling a NDA or changing a NDA will result in an airborne abort of a NDA connection). This is core functionality of CPDLC and must be supported by all CPDLC ground and airborne systems.

4.2.2.1.4 The CPDLC topology also permits another connection between an aircraft and a ground system. This connection is to facilitate the delivery of a DownStream Clearance (DSC) from a Downstream Data Authority (DDA). The DSC connection can only be established and terminated (except aborts) by the airborne system. An aircraft can have only one DSC connection at any one time. The DSC connection is independent of any CDA or NDA connection. The DDA and the NDA can be the same ground system, and a NDA and DSC connection to the same ground system are considered operationally valid. If an airborne system has a DSC connection with a ground system, and a CDA connection is established with that ground system, the airborne system must terminate the DSC connection. A CDA connection and DSC connection to the same ground system is not considered operationally valid. Operationally the DSC connection is expected to be of short duration (request and receipt of a downstream clearance). While the DSC connection is active, either the pilot (airborne system) or controller (ground system) can initiate CPDLC message exchanges. An airborne system does not need to support the DSC capability to be a conformant CPDLC configuration. A ground system must, at a minimum, recognize the request to establish a DSC connection (and can subsequently reject such a request) to have a conformant CPDLC configuration.

4.2.2.2 *Ground-Ground Link*

4.2.2.2.1 A given CPDLC ground system can establish a CPDLC link with another ground system for the purpose of forwarding a CPDLC message. The connection is essentially “one-shot” and is closed after receipt of confirmation from the receiving ground system This is a one way exchange of operational information. A ground system does not need to implement the capability to establish a ground-ground CPDLC link, but must at a minimum recognize the request to establish ground-ground CPDLC connection (and can subsequently reject such a request) to have a conformant CPDLC configuration.

4.2.2.3 *Illustration of Typical Topologies*

4.2.2.3.1 Figure 4.2-1 illustrates the CPDLC topology when an aircraft has a CDA and NDA connection. Ground connectivity is available for forwarding of CPDLC messages for transfer of data authority, as well as for delivery of downstream clearance information.

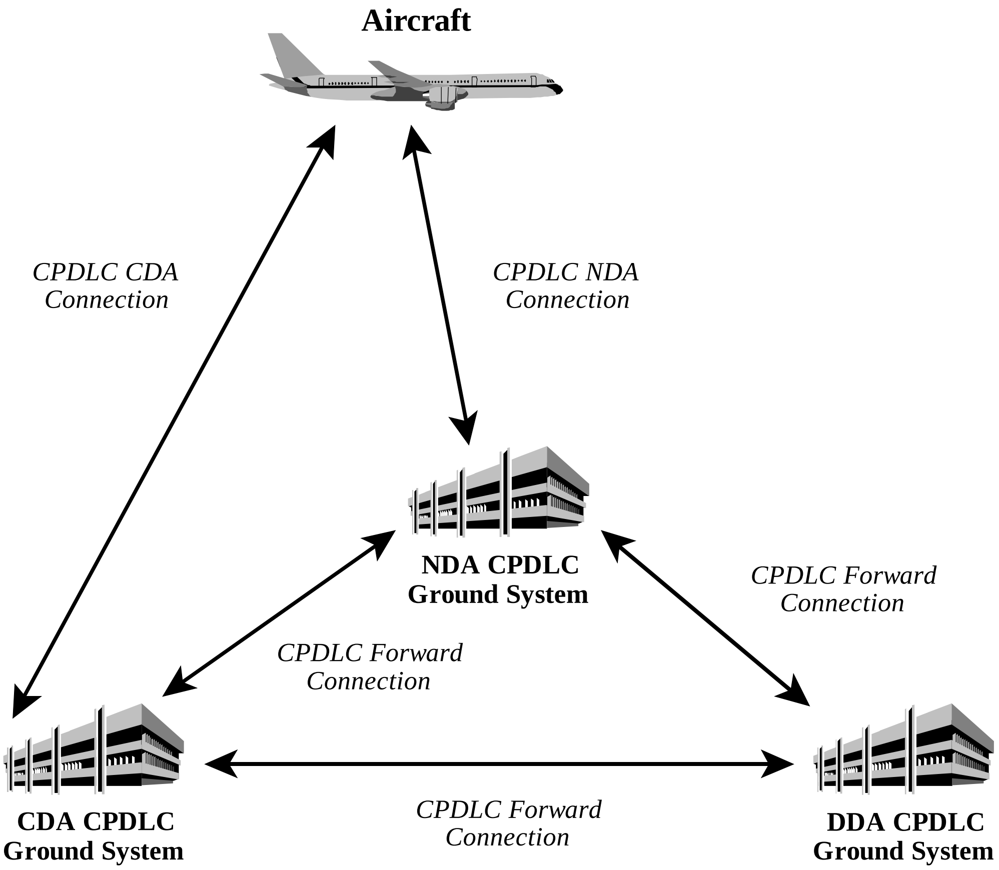


Figure 4.2-1. Aircraft With CDA and NDA Connection, Ground-Ground CPDLC Connection

4.2.2.3.2 Figure 4.2-2 illustrates the CPDLC topology when an aircraft has a CDA, NDA, and DSC connection. Ground connectivity is available for forwarding of CPDLC messages for transfer of data authority, but not for delivery of downstream clearance information.

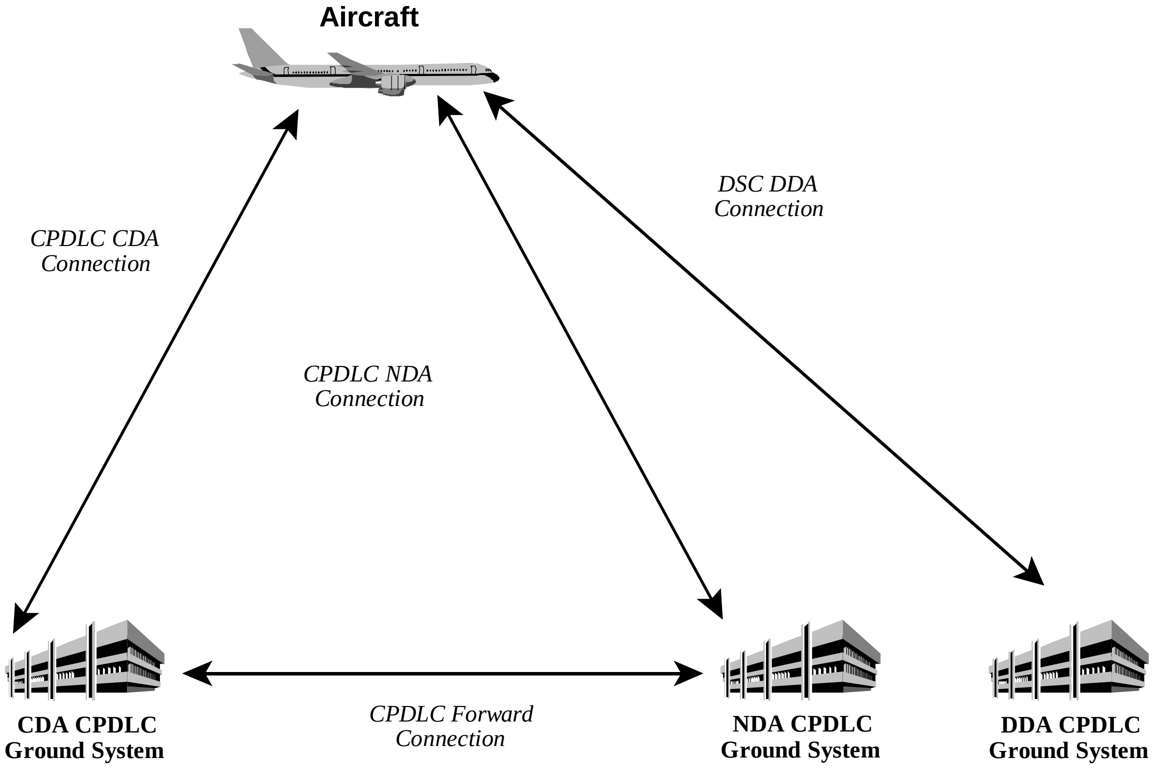


Figure 4.2-2. Aircraft With CDA, NDA and DDA Connection, No CDA-DDA Ground-Ground Connectivity

4.2.2.3.3 Figure 4.2-3 illustrates the CPDLC topology when an aircraft has a CDA, NDA, and DSC connection. Ground connectivity is not available, and the NDA and the DDA are the same ground system.

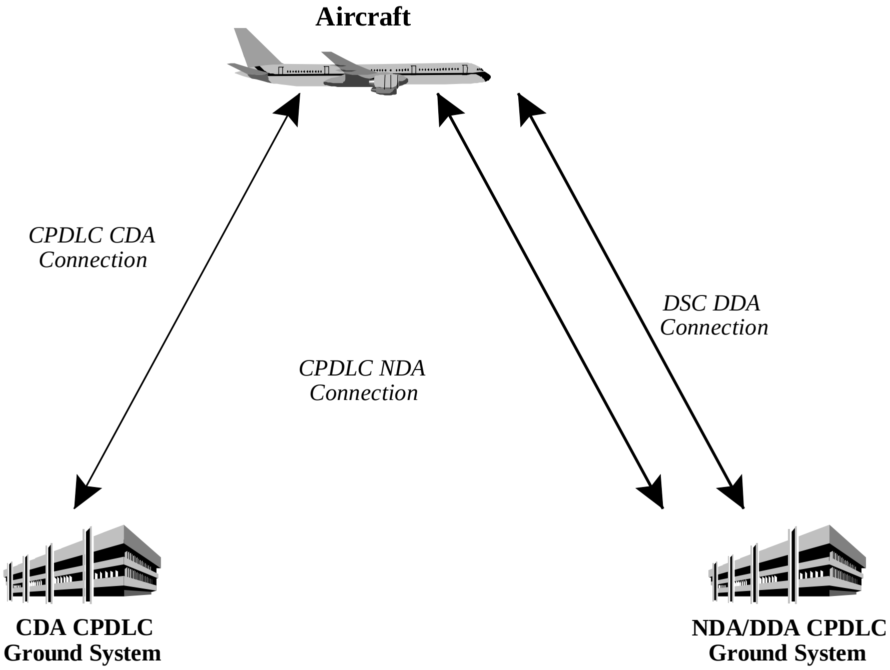


Figure 4.2-3. Aircraft with CDA, NDA, and DDA Connection, No Ground-Ground Connectivity

4.2.2.3.4 Figure 4.2-4 illustrates the CPDLC topology when an aircraft has a CDA link that contains more than one control authority and has a DSC connection. No NDA exists. There is no CPDLC ground-ground connectivity.

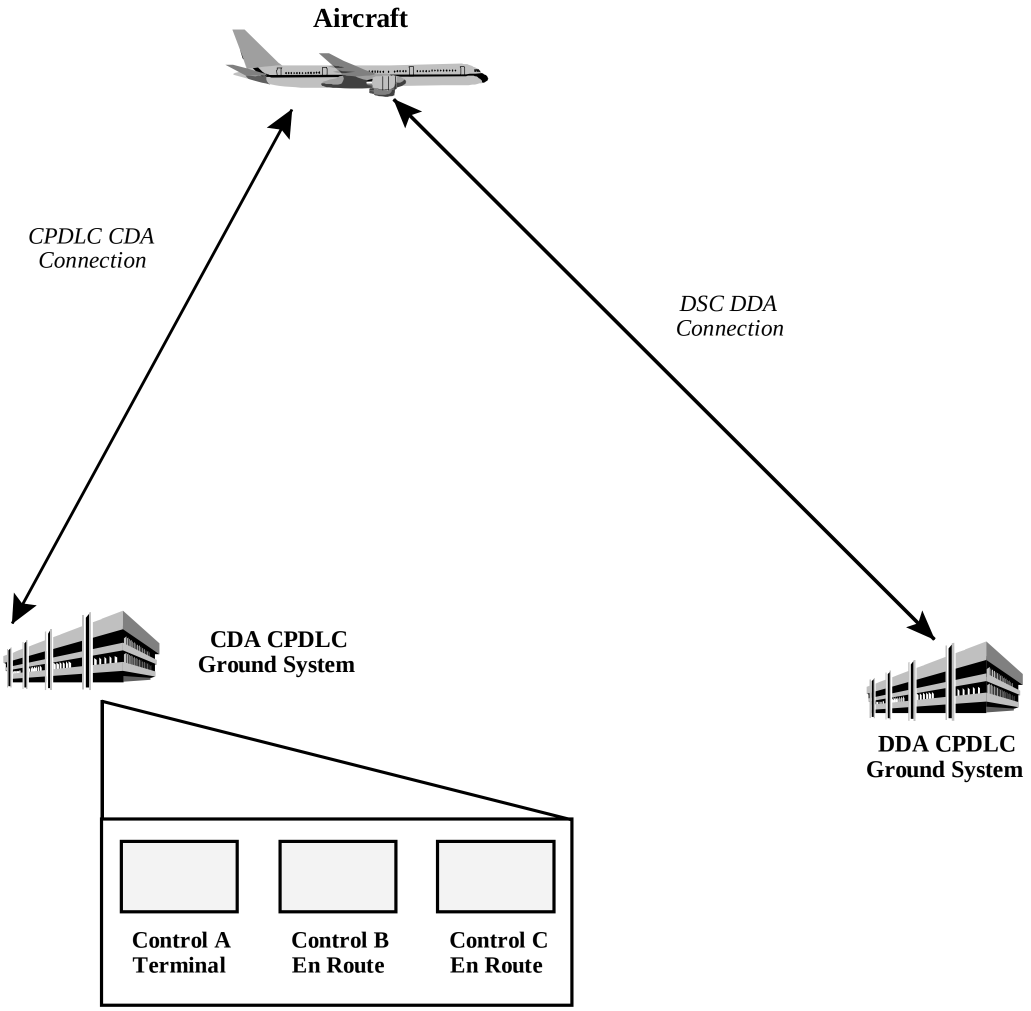


Figure 4.2-4. Aircraft With CDA Link Having More Than One Control Authority, DDA Link

4.2.3 Abstract Internal Architecture

4.2.3.1 *General*

4.2.3.1.1 The architectural model for the CPDLC Application conforms to [2]. One Application Entity (AE) is defined per CPDLC Application. Each AE contains an ATN Application Service Element (ASE), which is the communication element responsible for the ATN Application. The CPDLC SARPs specify the CPDLC AE and CPDLC-ASE.

4.2.3.2 *CPDLC-ASE Functionality*

4.2.3.2.1 The CPDLC application defines only one type of ASE, the CPDLC-ASE. The CPDLC-ASE has two variations: the CPDLC-air-ASE and the CPDLC-ground-ASE. The CPDLC-air-ASE in turn has two functional modes: CPDLC or DSC. Both of these modes are used exclusively in air-ground links. The CPDLC-ground-ASE also has two functional modes: CPDLC or DSC. The DSC mode is used exclusively in air-ground links. The ground CPDLC functional mode can in turn be either in a air-ground or ground-ground mode. A ground-ground CPDLC-ASE can be either an initiating or receiving ASE. The ground-ground CPDLC-ASE is used exclusively in the CPDLC-forward function. A CPDLC-ASE can act in only one mode at a time. For example a CPDLC-air-ASE cannot be both in DSC mode and CPDLC mode at the same time. The different modes of operation are depicted in Figure 4.2-5.

Figure 4.2-5. CPDLC-ASE Modes

4.2.3.2.2 The CPDLC-ASE is responsible for all aspects air-ground and ground-ground communication related to CPDLC. It encodes and decodes the PDUs. It maintains a state machine that only allows PDUs to be sent at an appropriate time, and detects if the peer application sends PDUs at an inappropriate time.

4.2.3.2.3 The CPDLC-ASE handles the protocol for the CPDLC application. If individual functions are not supported, as allowed by the subsetting rules (a conformant subset), the module will ensure that the protocol will handle the remaining subset of the full CPDLC functionality.

4.2.3.2.4 There is no architectural capability for multiple instances of the CPDLC-ASE within the same CPDLC AE. This implies that the CPDLC-ASE will generate and manage only one dialogue over the lifetime of the ASE.

4.2.3.3 CPDLC-User Functionality

4.2.3.3.1 In the model of the CPDLC, there is a module called the CPDLC-user. The functionality of the CPDLC-user is described in Section 2.3.7 of the CPDLC SARPs. Either the CPDLC-air-user or the CPDLC-ground-user is permitted to initiate the air-ground CPDLC service. Only the CPDLC-air-user is permitted to initiate the DSC service. Once an air-ground CPDLC or DSC dialogue is established either user can initiate the CPDLC message service. Only the CPDLC-ground-user can normally (non-abort conditions) initiate termination of a CPDLC dialogue. Only the CPDLC-air-user can normally (non-abort conditions) initiate termination of a DSC dialogue. Although there are requirements placed on the air-ground CPDLC-users, these tend to be the minimum required for interoperability. There is still a great deal of freedom in the method of implementation of this component.

4.2.3.3.2 The CPDLC-forward service is initiated by a CPDLC-ground-user. It is a ground-ground service. A CPDLC-forward dialogue is maintained just long enough for the initiator to receive confirmation on the forwarded CPDLC message. There is no concept of a two-way exchange of CPDLC messages in the forward service. The service (although technically confirmed) is operationally a one-shot, one CPDLC message operation. Although there are requirements placed on the ground-ground CPDLC-users, these tend to be the minimum required for interoperability. There is still a great deal of freedom in the method of implementation of this component.

4.2.3.4 *Product Architecture*

4.2.3.4.1 The CPDLC SARPs have defined an abstract model for the purposes of definition. That is, it has split the functionality between the ASE and the user and has defined an abstract interface between the two. It is strongly emphasized that there is no requirement on implementors to build the interface defined in the SARPs between the ASE and the user. If it is convenient, from an engineering perspective, to build an interface between two modules that embody the functionality of the ASE and the user, then the implementors are free to do so. However, if it is more convenient to build the system with interfaces in other places, then that is also acceptable. In testing a product to see if it conforms to the SARPs, no test can be made to test internal interfaces within the system.

4.2.3.4.2 The internal structure of the ATN ASE is not standardized across the air-ground applications; for example, CPDLC and CM are defined as a single module while ADS and FIS are defined as multiple modules.

4.2.4 Implementation Dependent Functionality

4.2.4.1 The CPDLC SARPs specify some of the requirements for the user, but leave a lot up to the implementor. There are no requirements that state how the user interface appears, how CPDLC interacts with the systems generating the CPDLC information, how CPDLC interacts with higher level functionality or with other applications such as CM or ADS. All this is implementation dependent.

4.2.4.2 The definition of the message elements composing a CPDLC message in the SARPs is quite open. It has not been possible to specify with the ASN.1 notation all the constraints and relationships between message elements. The rules for building a consistent CPDLC message are not checked by the CPDLC-ASE and have to be implemented by the CPDLC-users. There are a few constraints placed on message composition outlined in SARPs Section 2.3.7. These constraints are not checked by the sending or receiving ASE, but are requirements placed solely on the CPDLC-user.

4.2.5 Rationale for ASE / User Split

4.2.5.1 The rationale for the split in functionality between the CPDLC-ASE, the CPDLC-user, and the implementation dependent CPDLC functionality is as follows:

4.2.5.1.1 The ASE contains all functionality that is necessary to ensure the interoperability at the syntactic level. That is, two valid implementations of ASEs will be able to interact, passing data to each other in the correct order. They will be able to check the format of the data, ensure that it has been sent with the appropriate point in the dialogue, and also ensure that the peer ASE is behaving according to the requirements in the SARPs. Thus, the ASE ensures interoperability.

4.2.5.1.2 The CPDLC SARPs section 2.3.7 define some requirements for the users. These are the minimum user requirements necessary to ensure the semantic interoperability of the two peers. The user requirements explain how the data that is transported by the ASE should be interpreted. The user requirements explain how the following types of CPDLC/DSC dialogues operate:

a) CPDLC dialogue between a CPDLC-air user and the Current Data Authority (CDA) CPDLC-ground-user,

b) a CPDLC dialogue between a CPDLC-air user and the Next Data Authority (NDA) CPDLC-ground-user,

c) a DSC dialogue between a CPDLC-air user and the Downstream Data Authority (DDA) CPDLC-ground-user, and

d) a CPDLC dialogue between two CPDLC-ground-users.

4.2.5.1.3 Some care has been taken to ensure that the requirements are not over-specified. That is, they do not specify rules which are not absolutely essential to the syntactic and semantic interoperability of the CPDLC function. Requirements outside of the SARPs are considered implementation dependent functionality. The implementation dependent functionality can be built by different manufacturers in different ways, without affecting the interoperability between different implementations. Although not specified by the CPDLC SARPs, the implementation dependent functionality should be specified by individual product manufacturers or regional standards.

4.2.6 Inter-relationship with other ATN Applications

4.2.6.1 There is no direct interaction required between the CPDLC Application and the other ATN applications. The CPDLC application is a stand alone application which can be developed, certified, installed and operated completely independently from the other ATN Applications.

4.2.6.2 The pre-requisite for establishment of a CPDLC or DSC dialogue is the knowledge by the initiating peer of the name, address, and version number of the contacted peer. The Context Management (CM) Application is the means defined to exchange this information for air-ground dialogues. Thus, there is a technical relationship between the CPDLC and CM applications although is no direct interaction between the CM application and the CPDLC application. If the CM application is used to obtain application initiation information, the CM application must make this information available for use by the CPDLC air-ground application.

4.2.6.3 From a technical point of view, there is no relationship between the CPDLC application and other ATN applications such as ADS or ATS Interfacility Data Communication (AIDC). However, operationally they could be linked in a single service. For example, flight plan coordination information exchanges using AIDC, could be used to determine when to initiate a CPDLC dialogue. Information obtained from an ADS extended projected profile could be compared to the stored flight plan and the CPDLC application could be used to resolve any discrepancies. Such operational linkage of the ATN applications is beyond the scope of the CPDLC SARPs or guidance material.

4.2.7 Ground CPDLC Exchanges

4.2.7.1 Although all of the CPDLC operational requirements could technically be met using only air-ground CPDLC/DSC dialogues; ground-ground CPDLC message forwarding can be used to complement the air-ground capability. There are no specific operational requirements delineating CPDLC requirements that are air-ground vs. ground-ground or vise versa. The SARPs define a ground-ground CPDLC message forwarding capability to complement the basic air-ground CPDLC application in the following areas:

a) to forward CPDLC clearance messages from a downstream data authority to the current data authority for delivery to an aircraft, eliminating the need for a DSC air-ground link;

b) to allow the CDA to forward an aircraft CPDLC message request to the NDA or DDA, and notify the aircraft that the given message has been forwarded, eliminating the need for an aircraft to re-initiate the request with the data authority when it becomes the CDA.

c) CPDLC ground-ground capabilities can also be used in conjunction with the AIDC application to facilitate coordination between two ground systems.

4.2.8 **Dialogue Management**

4.2.8.1 The term "dialogue" refers to the end-to-end communication path provided by the Dialogue Service Provider. The Dialogue Service is described in detail in [2]. A dialogue is mapped directly onto a transport connection.

4.2.8.2 The dialogue service is used by the CPDLC-ASEs for the following purposes:

- establishment, graceful release, and abort of a dialogue

- transfer of data,

- support for quality of service and version number negotiation,

- application naming.

4.2.8.3 ***Optimization of the use of dialogues***

4.2.8.3.1 Establishment and termination of an air-ground (CPDLC or DSC) dialogue is always done explicitly by a CPDLC-user. There is no implicit dialogue establishment and termination as described in the ADS and FIS applications. Since a CPDLC dialogue can technically be initiated by either a CPDLC-air or a CPDLC-ground-user there are explicit instructions in the CPDLC SARPs for the situation when two CPDLC dialogues are established between an air-ground peer pair.

4.2.8.3.2 Establishment of a ground-ground CPDLC dialogue is always done explicitly by a CPDLC-user. There is no implicit dialogue establishment as described in the ADS and FIS applications. Dialogue termination is always implicit, done upon receipt of a confirmation of the ground forward request (completion of the CPDLC-forward service). Dialogue termination occurs whether or not the function was operationally successful. The CPDLC-forward service is a "one-shot" service, with no concept of maintaining a dialogue.

4.2.8.4 ***Dialogue Establishment***

4.2.8.4.1 In order to be able to initiate a CPDLC or DSC dialogue, some naming and addressing information have to be known from the dialogue initiator:

4.2.8.4.1.1 Establishment of air-ground dialogues:

a) Ground Initiation (CPDLC):

1) the ATN address of the airborne CPDLC system must be present somewhere in the ground data link communication system. Without this information, a CPDLC dialogue with a airborne CPDLC system can not be established.

2) the version number of the CPDLC protocol run on the airborne system must be available to the CPDLC-ground-user. The protocol version negotiation is not performed by the CPDLC Application Entities and the CPDLC-ground-user is responsible for checking the version compatibility before establishing a CPDLC dialogue.

b) Air Initiation (CPDLC and DSC):

1) the ATN address of the ground CPDLC system must be present somewhere in the airborne data link communication system. Without this information, a CPDLC or DSC dialogue with a ground CPDLC system can not be established.

2) the version number of the CPDLC protocol run on the ground system must be available to the CPDLC-air-user. The protocol version negotiation is not performed by the CPDLC Application Entities and the CPDLC-air-user is responsible for checking the version compatibility before establishing a CPDLC or DSC dialogue.

4.2.8.4.1.2 Establishment of ground-ground dialogues:

a) the ATN address of the ground CPDLC system must be present somewhere in the initiating ground data link communication system. Without this information, a CPDLC dialogue with another CPDLC ground system can not be established.

b) CM does not perform any address or version number exchange function for ground-ground dialogues, this information will have to acquired and made available to the CPDLC ground application by some other mechanism.

c) version number compatibility between two CPDLC ground systems is checked at the time a dialogue is initiated. (See Sections 2.3.3 and 2.3.5 of the CPDLC SARPs.)

4.2.8.4.2 A CPDLC or DSC dialogue can only be initiated by an explicit request by a CPDLC-user. There is no implicit CPDLC/DSC dialogue initiation. Section 4.7.2 provides several scenarios illustrating dialogue initiation. Table 4.2-1 shows the only ways in which a CPDLC/DSC dialogue can be initiated:

Table 4.2-1. CPDLC/DSC Dialogue Initiation

	Initiation Requirement	Type of Dialogue	Valid Source
1	CPDLC-start request	air-ground CPDLC dialogue	CPDLC-air-user CPDLC-ground-user
2	DSC-start request	air-ground DSC dialogue	CPDLC-air-user
3	CPDLC-forward request	ground-ground CPDLC dialogue	CPDLC-ground-user

4.2.8.4.3 The CPDLC SARPs outlines specific requirements for when a CPDLC/DSC dialogue can be initiated. During the time of the dialogue establishment (a start or forward request has been issued and the confirmation has not yet been received), no new request, other than an abort, can be issued by the CPDLC-user for a given peer connection.

4.2.8.4.4 **Crossing Start Conditions**

4.2.8.4.4.1 Since a CPDLC air-ground dialogue can be established by either the CPDLC-air-user or the CPDLC-ground-user there is a possibility for “simultaneous” starts that result in two CPDLC/DSC dialogues between given air and ground system pairs. This can happen in any of the following ways to an air-ground pair:

a) crossing starts resulting in two CDA dialogues,

b) crossing starts resulting in two NDA dialogues,

c) crossing starts resulting in a CDA and DSC dialogue, or

d) crossing starts resulting in a NDA and DSC dialogue.

Note.— Since only the aircraft can initiate a DSC, two crossing DSC starts cannot occur. Since the concept of an NDA only exists subsequent to a CDA dialogue in place, a CDA-NDA crossing start cannot occur. There is no concept of crossing starts in ground-ground CPDLC dialogues. If a given ground-ground pair both initiate the CPDLC-forward service such that two connections result, these connections are operationally both acceptable and independent.

4.2.8.4.4.2 If crossing starts result in two CDA dialogues being established between an air-ground pair, the CPDLC SARPs (section 2.3.7.4) requires that the CPDLC-air-user abort the connection that was established first. CDA mapping is maintained for the connection still in place. This scenario is illustrated in section 4.7.2 Figure 4.7-7 of this chapter.

4.2.8.4.4.3 If crossing starts result in two NDA dialogues being established between an air-ground pair, the CPDLC SARPs (section 2.3.7.4) requires that the CPDLC-air-user abort the connection that was established first.

4.2.8.4.4.4 If crossing starts result in a CDA dialogue and a DSC dialogue being established between an air-ground pair, the CPDLC SARPs section 2.3.7.4 requires that the CPDLC-air-user end the DSC connection. The CPDLC-air-user may choose to end the DSC connection gracefully using the DSC-end service, or abruptly using the CPDLC-user-abort service. There is no technical problem with a CDA and DDA being the same ground system, however operationally this does not make sense, thus the DSC connection should be terminated, although the termination need not be abrupt. This also implies that operationally it does not make sense for an aircraft to initiate a DSC dialogue with a ground system that is already its CDA, even though the CPDLC SARPs do not technically prohibit the CPDLC-air-user from doing so. The scenario where the CPDLC-air-user aborts the DSC dialogue is illustrated in section 4.7.2 Figure 4.7-9 of this chapter.

4.2.8.4.4.5 If a crossing start results in a NDA dialogue and a DSC dialogue being established between an air-ground pair, this is considered operationally acceptable. The CPDLC SARPs do not contain any constraints or requirements concerning this situation. A NDA may well also be the DDA. A NDA connection cannot be used for any CPDLC message exchange, whereas a DSC dialogue can be used for CPDLC message exchange. This scenario is illustrated in section 4.7.2 Figure 4.7-10 of this chapter.

4.2.8.4.5 Summary of CPDLC-Start Request Requirements

4.2.8.4.5.1 The SARPs permit a CPDLC-air-user to issue a CPDLC-start request with any ground system if the CPDLC-air-user currently does not have any CPDLC dialogues. Ground acceptance of any CPDLC-start request will be determined by operational or procedural requirements. For example, a given ground system may not permit establishment of air initiated CPDLC dialogues.

4.2.8.4.5.2 Once any CPDLC dialogue is established, the SARPs constrains the CPDLC-air-user to only being permitted to issue a CPDLC-start request with an NDA. As in the preceding paragraph, ground acceptance of any air initiated CPDLC-start request will be determined by operational or procedural requirements.

4.2.8.4.5.3 The SARPs do not constrain a CPDLC-ground-user from issuing a CPDLC-start request with any aircraft at any time other than the time of dialogue establishment with a given aircraft peer. There are specific requirements in the SARPs that totally determine when a CPDLC-air-user must accept or reject a CPDLC-start request. Since any ground system could technically establish a CPDLC air-ground connection, operational and procedural rules may constrain CPDLC-ground initiation. The first CPDLC connection that is established determines the CDA, and thus effectively locks out any other ground system (legitimate or not) from establishing any CPDLC connection other than a NDA CPDLC connection with a given aircraft.

4.2.8.4.5.4 The mapping of CDA, NDA, and DDA designations to ground systems is an CPDLC-air-user function.

4.2.8.4.6 **Summary of DSC-Start Request Requirements**

4.2.8.4.6.1 The SARPs permit a CPDLC-air-user to issue a DSC-start request with any ground system if the CPDLC-air-user currently does not have any DSC dialogues. Ground acceptance of any DSC-start request will be determined by operational or procedural requirements. For example, a given ground system may not support the DSC function. (It must support a stub of the function to enable rejection of the request for a DSC dialogue.)

4.2.8.4.7 An aircraft can have only one DSC dialogue at a time. That is, the SARPs do not permit a CPDLC-air-user to establish a DSC dialogue with two ground systems concurrently; nor do they permit a CPDLC-air-user to have more than one DSC dialogue with the same ground system.

4.2.8.4.8 Although the SARPs do not prohibit the CPDLC-air-user from opening a DSC dialogue with its CDA (provided another DSC connection is not already established, since there can only be one DSC connection for a given CPDLC-air-user), the SARPs do require a CPDLC-air-user to terminate a DSC dialogue with a ground system that is its CDA. So, if such a DSC dialogue were opened with a CDA, the CPDLC-air-user would have to promptly abort such a connection. Operationally this configuration of a ground system simultaneously being a CDA and a DDA is contradictory.

4.2.8.4.9 The SARPs do not constrain in any way a CPDLC-air-user from opening a DSC dialogue with a ground system that is its NDA (provided another DSC connection is not already established, since there can only be one DSC connection for a given CPDLC-air-user). Operationally, the configuration is both possible and logical. The NDA connection cannot be used for operational message exchange, so communication with an NDA that is also the DSC must be done through the DSC connection.

4.2.8.4.9.1 The SARPs do not permit a CPDLC-ground-user to issue a DSC-start request.

4.2.8.4.10 **Summary of CPDLC-forward Request Requirement**

4.2.8.4.10.1 The SARPs do not require that a ground system have the capability to issue a CPDLC-forward request.

4.2.8.4.10.2 The SARPs permit a CPDLC-user to issue a CPDLC-forward request, if CPDLC-forward request capable, with any other CPDLC ground system at any time other than the time of dialogue establishment with a given ground system peer. Requirements dictating rejection of ground-ground CPDLC dialogue establishment are specified in the SARPs. Any requirements determining acceptance of a ground-ground CPDLC dialogue will be determined by operational and procedural rules outside of the SARPs. All CPDLC ground systems must have the technical capability of receiving a CPDLC-forward request/indication, and issuing a response.

4.2.8.5 *Dialogue Termination*

4.2.8.5.1 An air-ground CPDLC/DSC dialogue is not terminated unless specifically commanded to do so by either an orderly release, or abruptly by an abort condition. A CPDLC/DSC air-ground dialogue is never implicitly released for any reason (e.g. due to lack of CPDLC message traffic). A CPDLC ground-ground dialogue is never maintained beyond the receipt of a CPDLC-forward confirmation, and can only be terminated prior to this by an abort condition. Table 4.2-2 shows the only ways in which a CPDLC/DSC dialogue can be terminated:

Table 4.2-2. CPDLC/DSC Dialogue Termination

	Termination Requirement	Type of Dialogue	Valid Source
1	CPDLC-start response/confirmation *Result* "rejected"	air-ground CPDLC dialogue	CPDLC-air-user CPDLC-ground-user
2	DSC-start response/confirmation *Result* "rejected"	air-ground DSC dialogue	CPDLC-ground-user
3	CPDLC-forward response/confirmation	ground-ground CPDLC dialogue	CPDLC-ground-user
4	D-START response/confirmation *Result* "rejected"	ground-ground CPDLC dialogue	CPDLC-ground-ASE
5	CPDLC-user-abort D-U-ABORT	any CPDLC or DSC dialogue	CPDLC-air-user CPDLC-ground-user
6	CPDLC-provider-abort D-U-ABORT	any CPDLC or DSC dialogue	CPDLC-air-ASE CPDLC-ground-ASE
7	D-ABORT	any CPDLC or DSC dialogue	CPDLC communication service provider
8	CPDLC-end response/conformation *Result* "accepted"	air-ground CPDLC dialogue	CPDLC-air-user
9	DSC-end response/confirmation response/conformation *Result* "accepted"	air-ground DSC dialogue	CPDLC-ground-user

4.2.8.5.2 **Summary of CPDLC/DSC Start Rejection Requirements**

4.2.8.5.2.1 A CPDLC/DSC dialogue can be terminated during the time that it is still being established by either an abort condition (outlined below) or by a CPDLC-start response set to the abstract value "rejected". The SARPs specifically (but not totally) constrain the CPDLC start response rejection requirements. The SARPs permit/require that a request to start a CPDLC/DSC dialogue be rejected (non-abort conditions) for any of the following reasons:

a) the source for the request is invalid (e.g. not the CDA, when a dialogue is in place, not the designated NDA),

b) the Class of Communication is considered insufficient by a CPDLC-ground-user,

c) the CPDLC-ground system does not support air initiated CPDLC dialogues,

d) the CPDLC-ground system does not support the DSC function,

e) the CPDLC dialogue is a ground-ground dialogue. (The forward function is "one-shot" and a dialogue is always terminated upon receipt of the CPDLC-confirmation.)

4.2.8.5.3 Summary of Abort Conditions

4.2.8.5.3.1 The SARPs permit/require that a CPDLC/DSC dialogue can be terminated by an abort at any time when any of the following conditions occurs;

a) the underlying system detects an error or fails (D-ABORT, CPDLC-Provider-Abort),

b) the CPDLC-ASE detects an error or fails (D-U-ABORT, CPDLC-Provider-Abort),

c) technical timers expire (D-U-ABORT, CPDLC-Provider-Abort),

d) the CPDLC-user detects an error requiring an abort, the CPDLC-user fails, (CPDLC-User-Abort),

e) CPDLC/DSC dialogue management requires an abort (CPDLC-User-Abort), or

f) the CPDLC-user wishes to abort (CPDLC-User-Abort).

4.2.8.5.4 Summary of CPDLC-End Requirements

4.2.8.5.4.1 An established CPDLC dialogue is terminated (non-abort conditions) when a CPDLC-end confirmation is received with the value "accepted". The SARPs only allow a CPDLC-ground-user to issue a CPDLC-end request. The SARPs do have any requirements on when a CPDLC-ground-user must issue a CPDLC-end-request.

4.2.8.5.4.2 The SARPs very specifically (but not totally) constrain the CPDLC-air-user CPDLC-end response requirements. The CPDLC-air user response requirements are based on the following criteria:

a) source of the CPDLC-end indication (only the CDA is valid),

b) errors in any CPDLC message contained in a CPDLC-end indication *CPDLC Message* parameter,

c) any CPDLC message that the CPDLC-air user has sent requiring a response, for which a response has not been received, and

d) response requirements for any CPDLC message contained in CPDLC-end indication *CPDLC Message* parameter.

4.2.8.5.4.3 **Summary of DSC-end Requirements**

4.2.8.5.4.4 An established DSC dialogue is terminated (non-abort conditions) when a DSC-end confirmation is received with the value "accepted". The SARPs permit only the CPDLC-air-user to invoke the DSC-end request. The SARPs require the CPDLC-air-user to terminate a DSC dialogue in two situations:

a) when the DDA becomes a CDA (the CDA CPDLC dialogue is established), and

b) prior to issuing any other DSC-start request. (A given aircraft can only have one DSC dialogue at a time.)

4.2.8.5.4.5 Although, no further constraints are placed by the SARPs on the CPDLC-air-user DSC-end request invocation, the DSC is operationally envisaged as having "short term" duration. That is, a request for a downstream clearance is issued to the DDA and the DSC dialogue is terminated upon the receipt of the DSC clearance.

4.2.8.5.4.6 The SARPs very specifically (but not totally) constrain the CPDLC-ground-user DSC-end response requirements. The CPDLC-ground user response requirements are based on the following criteria:

a) errors in any CPDLC message contained in a DSC-end indication *CPDLC Message* parameter,

b) any CPDLC message that the CPDLC-ground user has sent requiring a response, for which a response has not been received, and

c) response requirements for any CPDLC message contained in DSC-end indication *CPDLC Message* parameter.

4.2.9 **Protocol Monitoring**

4.2.9.1 ***General***

4.2.9.1.1 The CPDLC-ASE controls that the protocol is correctly handled by the peer and can be correctly operated locally. The generation and transmission of expected responses are monitored by the peer CPDLC-ASE.

4.2.9.1.2 In the case a service provider timer expires or if an exception is detected, the ASE will abort the dialogue in place will be terminated. Active CPDLC-users will be informed (if possible) of the abort by a CPDLC-provider-abort indication which includes a reason for the abort. These cases are described in the following sections.

4.2.9.2 *Exception Handling*

4.2.9.2.1 CPDLC Service Provider Timers

4.2.9.2.1.1 If the CPDLC-start, DSC-start, or CPDLC-forward confirmation is not received by the requesting CPDLC-ASE within a locally specified period of time (CPDLC SARPs section 2.3.5 recommends 6 minutes), the dialogue being established is aborted. Both CPDLC-users are informed of this situation by a CPDLC-provider-abort indication with a reason of “timer-expired”. This timer is a service provider timer and is not directly connected to any operational timers set, except that it should be sufficiently larger than any operational timer to prevent “nuisance” timer-expiration aborts.

4.2.9.2.1.2 Since the SARPs prohibit the initiating user from initiating any other (non-abort) CPDLC or DSC service once a start or forward service request is issued until the corresponding confirmation is received, the SARPs prohibit the inclusion of any CPDLC message in an accepted start response. This is required to minimize the time required to fully establish the dialogue, since CPDLC operational responses usually require human input.

4.2.9.2.1.3 CPDLC (unlike ADS or FIS) does not have any “inactivity timers”. A CPDLC dialogue is never terminated due to inactivity, but must be explicitly terminated as outlined in section 4.2.8.5 above.

4.2.9.2.1.4 CPDLC (unlike ADS or FIS) does not have any “end timers”. A CPDLC dialogue is never terminated automatically due to lack of receipt of a D-END confirmation. Since nearly all of CPDLC responses involve human responses (unlike ADS or FIS), response times are governed locally by required operational response times rather than technical timers.

4.2.9.2.2 Protocol Errors

4.2.9.2.2.1 Unrecoverable System Error

4.2.9.2.2.2 The unrecoverable system error is intended to cover cases where a fault causes a system lockup or the system to become unstable (e.g., the system has insufficient memory). If the system is capable, the CPDLC application will abort the connection and, if able, inform both the local and peer users of the situation with the CPDLC-provider-abort reason “undefined-error”.

4.2.9.2.2.3 The unrecoverable system error processing is written as a recommendation in the SARPs instead of a requirement, as it is recognized that depending on the nature of the error in the system, it may not be possible to regain control in order to either perform an abort, or inform the users of the abort situation.

4.2.9.2.3 Invalid PDU

4.2.9.2.3.1 An invalid Protocol Data Unit (PDU) is defined as a PDU that cannot be decoded or that is not received when it is expected. For example, a PDU that somehow becomes garbled would be an invalid PDU. There are two cases for handling invalid PDUs, the first requiring both CPDLC-users be informed, if active, because a dialogue is either in place or being established (thus D-ABORT is invoked), and the second requiring only the local user be informed, if active, because a dialogue is in the process of being closed (D-ABORT is not invoked.) An abort due to an invalid PDU is always invoked by a receiving CPDLC-ASE. The abort issued to the CPDLC-user contains the CPDLC-provider abort reason “invalid-PDU”.

4.2.9.2.4 Not Permitted PDU

4.2.9.2.4.1 A not permitted PDU is defined as a PDU that arrives when the CPDLC-ASE is in a state that does not allow that PDU. For example, receiving a *User Data* parameter in a D-START confirmation when the start has been accepted is not permitted. This does not mean that the PDU cannot be decoded; and in fact it may well be a valid PDU. There are two cases for handling not permitted PDUs, the first requiring both CPDLC-users be informed, if active, because a dialogue is either in place or being established (thus D-ABORT is invoked), and the second requiring only the local user be informed, if active, because a dialogue is in the process of being closed (D-ABORT is not invoked.) An abort due to a not permitted PDU is always invoked by a receiving CPDLC-ASE. The abort issued to the CPDLC-user contains the CPDLC-provider abort reason “not-permitted-PDU”.

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

4.2.9.2.5.1 CPDLC-ASEs should never receive a D-START confirmation with the D-START *Result* parameter having the abstract value “rejected (transient)” nor a D-START *Reject Source* parameter having the abstract value “DS provider”. The D-START *Result* parameter is set by the CPDLC-ASEs, and is if SARPs compliant is either “rejected (permanent)” or “accepted”. The D-START *Reject Source* parameter is set by the dialogue service provider, and is normally “DS User”. The D-START *Result* abstract value “rejected (transient)” or the D-START *Reject Source* abstract value “DS provider” is indicative of a communication failure. Therefore, an abort due to unexpected D-START confirmation values is always invoked by the receiving CPDLC-ASE and a CPDLC-Provider-Abort is indicated to the local user, if active, containing the CPDLC-provider abort reason “communication-service-error”. A D-ABORT is not invoked since there is inherently no connection due to the communication error condition.

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

4.2.9.2.6.1 The ATN Sub-Volume 1 [4] dictates the values used for application service priority and Residual Error Rate (RER) quality of service parameters for all ATN applications. These values must be adhered to so proper levels of flight safety and performance are maintained. For CPDLC, the values used are “high priority flight safety messages” for application service priority and “low” for RER quality of service. If these values are not present (as checked in sections 2.3.5.4.6 and 2.3.5.6.6 of the CPDLC SARPs), then the CPDLC-ASE will abort with reason “invalid-QOS-parameter” and the abstract value of “provider” for the *Originator* parameter value. The abstract value “provider” is used since this is not a user-invoked abort. The receiving CPDLC-ASE will abort and issues a D-ABORT to the peer ASE containing the reason “invalid-QOS-parameter”, which in turn is provided to the peer user in a CPDLC-provider-abort indication . (In this situation the CPDLC-user of the CPDLC-ASE receiving the D-START indication is never active and therefore a CPDLC-provider-abort indication is not invoked by the receiving ASE.)

4.2.10 **Version Number Negotiation**

4.2.10.1 ***Version Number Negotiation (Air-Ground)***

4.2.10.1.1 The CPDLC SARPs specify the operation of version 1 of the CPDLC application. In the future, it can be anticipated that further CPDLC operational requirements will emerge (e.g. additional CPDLC messages), and therefore there will be a need to develop additional versions of the CPDLC SARPs.

4.2.10.1.2 The version number is a value inherent to the CPDLC-ASE and is not provided by the CPDLC-users.

4.2.10.1.3 When the CM application exchanges application names and addresses it also exchanges the associated supported version numbers of the air-ground applications. The CPDLC-user initiating a air-ground CPDLC or DSC dialogue must check whether the ASE implemented is compatible with one of the versions implemented on the peer system. A dialogue must not be initiated if the peer ASE version numbers are not compatible.

4.2.10.1.4 The CPDLC-forward function is different. There is no CM supporting ground-ground applications, and version number are passed during CPDLC-forward connection initiation. The receiving CPDLC-ground application must check the version number, and ensure that the version numbers are compatible. If the receiving CPDLC-ground system cannot support the requested version, it must reject the connection and supply the initiating CPDLC-ground system its version number.

4.3 Functionality of Services

4.3.1 Introduction

4.3.1.1 This section describes the information provided by the ASEs to the CPDLC-users and then the information required by the ASEs from the CPDLC-users. Then, it considers CPDLC services in turn and provides an overview of the data flow expected in normal operations within the ASE which handles the service primitives.

4.3.2 Concepts

4.3.2.1 Users of the CPDLC service are termed *CPDLC-ground-user* and *CPDLC-air-user* or just *CPDLC-user* when it applies to both air and ground. The CPDLC-user represents the operational part of the CPDLC system. It is either the final end-user (e.g. a crew member or controller) or an automated system. The CPDLC-user that initiates a CPDLC air-ground or ground-ground service is termed the *calling* CPDLC-user or *initiator*. The CPDLC-user that the initiator is trying to contact is termed the *called* CPDLC-user or *responder*.

4.3.2.2 This section considers each of primitives composing the CPDLC Service. The primitives are grouped according to the services they provide:

a) starting a CPDLC/DSC dialogue

    1) CPDLC-start service

    2) DSC-start service

    3) CPDLC-forward service

b) CPDLC message exchange

c) CPDLC-message service

d) ending a CPDLC/DSC dialogue

    1) CPDLC-end service

    2) DSC-end service

e) aborting a CPDLC/DSC dialogue

    1) CPDLC-user-abort service

    2) CPDLC-provider-abort service

4.3.3 **CPDLC Service Parameters**

4.3.3.1 ***Called Peer Identifier***

4.3.3.1.1 During CPDLC air-ground dialogue establishment (CPDLC-start request), the CPDLC-user must supply the identification of the called peer. The *Called Peer Identifier* is required for dialogue addressing purposes. When the dialogue is air initiated, the *Called Peer Identifier* is a *Facility Designation*. When the dialogue is ground initiated, the *Called Peer Identifier* is an *Aircraft Address*.

4.3.3.1.2 The *Called Peer Identifier* must be made available to the CPDLC-air-user when air initiated. The *Called Peer Identifier* must be made available to the CPDLC-ground-user when ground initiated.

4.3.3.1.3 See *Facility Designation* and *Aircraft Address* for the format of the *Called Peer Identifier*.

4.3.3.2 ***Calling Peer Identifier***

4.3.3.2.1 During CPDLC air-ground dialogue establishment (CPDLC-start request), the CPDLC-user must supply its identification. The *Calling Peer Identifier* is required to allow to the peer-user to identify its calling peer-user. When the dialogue is air initiated, the *Calling Peer Identifier* is an *Aircraft Address*. When the dialogue is ground initiated, the *Calling Peer Identifier* is a *Facility Designation*.

4.3.3.2.2 See *Facility Designation* and *Aircraft Address* for the format of the *Calling Peer Identifier*.

4.3.3.3 ***Facility Designation***

4.3.3.3.1 During DSC dialogue establishment (DSC-start request) the CPDLC-air-user must supply the *Facility Designation* of the called peer. The *Facility Designation* is required for dialogue addressing purposes. The *Facility Designation* must be made available to the CPDLC-air-user.

4.3.3.3.2 The *Facility Designation* parameter can be four to eight characters. Four characters are used to identify a particular facility, such as “ZYVR” for Vancouver. Up to eight characters may be used to identify a particular system within a facility; i.e. “ZYVRA123” may be the address for Vancouver en-route.

4.3.3.4 ***Aircraft Address***

4.3.3.4.1 During DSC dialogue establishment (DSC-start request) the CPDLC-air-user must supply its *Aircraft Address*. The *Aircraft Address* is required for indication to the peer-ground-user.

4.3.3.4.2 The *Aircraft Address* is the 24 bit aircraft address. The ICAO fight plan can contain the aircraft address in the remarks field, field 18.

4.3.3.5 *Called Facility Designation*

4.3.3.5.1 During CPDLC ground-ground dialogue establishment (CPDLC-forward request), the initiating CPDLC-ground-user must supply the identification of the called ground peer. The *Called Facility Designation* is required for dialogue addressing purposes.

4.3.3.5.2 The *Called Facility Designation* must be made available to the CPDLC-ground-user of the initiating ground system.

4.3.3.5.3 See *Facility Designation* for the format of the *Called Facility Designation.*

4.3.3.6 *Calling Facility Designation*

4.3.3.6.1 During CPDLC ground-ground dialogue establishment (CPDLC-forward request), the initiating CPDLC-ground-user must supply its identification. The *Calling Facility Designation* is required to allow a peer-user to identify its calling peer-user.

4.3.3.6.2 See *Facility Designation* for the format of the *Calling Facility Designation.*

4.3.3.7 *Class Of Communication Service*

4.3.3.7.1 The *Class of Communication* may be provided by the CPDLC-user initiating a CPDLC or DSC dialogue (CPDLC-start request, DSC-start request, or CPDLC-forward request). Technically the user is not required to supply the *Class of Communication.* However, operational procedures beyond the SARPs requirements may dictate when and what value of *Class of Communication* is supplied by the CPDLC-user.

4.3.3.7.2 The *Class of Communication* is a means for the user to indicate the required performance in terms of end-to-end transit delay. The *Class of Communication* provided by the CPDLC-user is supplied to the Transport Service Provider (TSP) when the connection is established. Values for these transit delays are given in [4]. The *Class of Communication* is not guaranteed nor a degradation of the provided class is indicated to the users. It is the responsibility of the application to determine the actual transit delay achieved by local means such as time stamping.

4.3.3.7.3 If the CPDLC-user does not require a particular *Class of Communication*, the *Class of Communication* parameter may be left blank, indicating no routing preference. This means Class of Communication is chosen by the dialogue service provider. The value supplied by dialogue service provider is then indicated to the receiving user.

4.3.3.7.4 There is no negotiation of the Class of Communication between air and ground CPDLC-users. However, the ground-user may have operational requirements dictating the minimum acceptable Class of Communication. Therefore the *Class of Communication* parameter is always indicated to the receiving user and can be used as a basis for a CPDLC-ground-user to reject the request to start a CPDLC or DSC dialogue. Although the *Class of Communication* is also always indicated to a CPDLC-air-user, the CPDLC-air-user cannot reject the request for a CPDLC dialogue based on the Class of Communication value.

4.3.3.7.5 Although there is no negotiation of the Class of Communication, if a CPDLC-ground-user rejects the CPDLC-air-user request for a CPDLC dialogue based on the indicated *Class of Communication* parameter value, the CPDLC-ground-user is free to subsequently initiate a CPDLC dialogue with the required Class of Communication. The CPDLC-air-user is also free to re-initiate the CPDLC-start request with a different Class of Communication value.

4.3.3.7.6 In the case where the CPDLC-ground-user reject a request for a DSC dialogue based on the indicated *Class of Communication* parameter value, it is incumbent on the CPDLC-air-user to re-initiate the DSC dialogue with a different Class of Communication.

4.3.3.7.7 The *Class of Communication* is used to implicitly indicate that the routing class is ATSC.

4.3.3.8 *CPDLC Message*

4.3.3.8.1 The *CPDLC Message* parameter contains a CPDLC message provided by the user of the CPDLC service. When invoking the CPDLC-start request, DSC-start request, CPDLC-end request, or DSC-end request primitive, the CPDLC protocol allows the user the choice of whether or not to supply this parameter. When invoking the CPDLC-forward and CPDLC-message request primitive, the CPDLC protocol requires that the user supply a CPDLC message. When invoking CPDLC-end or DSC-end service response primitive, the CPDLC protocol allows the user the choice of whether or not to supply this parameter. Whenever the *CPDLC Message* parameter is supplied by a user it is always indicated/confirmed to the receiving user.

4.3.3.8.2 User requirements in Section 2.3.7 of the CPDLC SARPs go beyond those in the communication protocol and mandate the provision of a CPDLC message in the CPDLC-end and DSC-end *CPDLC Message* parameter under specified conditions.

4.3.3.8.3 Furthermore, operational and procedural requirements may further constrain when a CPDLC message must/must not be supplied by the user of a service, and what particular message is supplied under given conditions. These requirements are outside the scope of the SARPs.

4.3.3.8.4 Whenever the provision of the *CPDLC Message* parameter is optional according to the protocol in Section 3 (i.e., a “U”), the receiving ASE does not check whether or not a message is present.

4.3.3.9 *Reject Reason*

4.3.3.9.1 The *Reject Reason* parameter contains a CPDLC message to be provided by the user of the CPDLC-start response or DSC-start response services only to indicate an operational reason for the rejection of the requested service. The parameter is supplied by the CPDLC-user receiving a CPDLC-start indication or DSC-start indication if and only if the receiving user is rejecting the requested start. The CPDLC protocol does not permit a CPDLC message to be included when a request to start a dialogue is accepted. The receiving ASE checks for the inclusion of a CPDLC message in both instances and aborts if the conditions stated above are not met. Whenever the *CPDLC Message* parameter is validly supplied by a user it is always confirmed to the receiving user.

4.3.3.10 *Result*

4.3.3.10.1 The *Result* parameter is used by the CPDLC-user to indicate acceptance or rejection of the request by the peer user to start or terminate a CPDLC or DSC dialogue. The CPDLC-user must always supply the *Result* parameter when invoking the CPDLC-start response, the DSC-start response, the CPDLC-end response or the DSC-end response service primitives. The CPDLC-user supplied *Result* parameter can have one of two values: “accepted” or “rejected”. A valid *Result* parameter is always confirmed to the peer user.

4.3.3.10.2 Section 2.3.7 places some constraints on the CPDLC-user supplied *Result* parameter value.

4.3.3.10.3 The *Result* parameter is confirmed to the initiating CPDLC-forward user. In the CPDLC-forward service the parameter is always set to “rejected” by the receiving CPDLC-ASE, and is never supplied by the receiving CPDLC-user.

Note.— The D-START Result parameter can have two abstract values that indicate dialogue rejection “rejected (transient)” or “rejected (permanent)”. The CPDLC-start service only uses the abstract value “rejected”. The CPDLC-ASE always maps the “rejected” value received in a CPDLC-start service response to the D-START “rejected (permanent)” value. The CPDLC application does not use the “rejected (transient)” value.

4.3.3.11 *ASE Version Number*

4.3.3.11.1 Since the CM version number negotiation only applies to air-ground dialogues, version number “negotiation” must be performed by the CPDLC application for ground-ground dialogues. The CPDLC version number is an intrinsic part of the ASE and does not need to be supplied by the CPDLC-ground-user. However the *ASE Version Number* parameter must be confirmed to the initiating CPDLC-ground-user whenever the receiving CPDLC-ground-ASE rejects the CPDLC-forward request due to version number incompatibility. The *Version Number Parameter* has an abstract integer value between 1 and 255.

4.3.3.12 *Reason*

4.3.3.12.1 The *Reason* parameter contains the reason for an abort. When a CPDLC abort is issued by a CPDLC-user, the user can supply an abort reason. When a CPDLC abort is initiated by a CPDLC-ASE, the ASE must indicate the abort reason to its CPDLC-.user, if able, if the user is an active user. Any abort reason received by a CPDLC-ASE from the dialogue service (in a D-ABORT or a D-U-ABORT) must indicated to its CPDLC-.user, if the user is an active user. The *Reason* parameter conforms to either the ASN.1 abstract syntax

CPDLCUserAbortReason, or to the ASN.1 abstract syntax CPDLCProviderAbortReason as specified in the CPDLC SARPs.

4.3.4 **Mode Explanation**

4.3.4.1 A CPDLC-user must be able to distinguish messages received over a DSC link from those received over a CPDLC link [3]. To enable this, two mode of air-ground CPDLC operation have been defined by the CPDLC SARPs. These air-ground modes of CPDLC operation are CPDLC and DSC. The mode is defined in the ASN.1. The mode defaults to "cpdlc" and must explicitly be set to "dsc" for DSC dialogues. The mode can only be set to "dsc" by the CPDLC-air-ASE upon receipt of a DSC-start request from the CPDLC-air-user. The mode is included as part of the StartDownMessage AircraftAPDUs. (Since DSC is only air initiated, the CPDLC-ground-ASE does not establish any mode of operation when initiating a CPDLC air-ground dialogue.) For air-ground dialogues, upon receipt of a D-START indication, the CPDLC-ground-ASE checks the mode contained in the StartDownMessage Application Protocol Data Unit (APDU). If the mode is "cpdlc" a CPDLC-start indication is provided, if the mode is "dsc" a DSC-start indication is provided. (Any mode value that is neither "dsc" nor "cpdlc" is invalid and causes an abort.) The mode is also checked by the CPDLC-ASE during CPDLC-end and DSC-end processing to ensure validity of the end indication/confirmation. Whenever a CPDLC/DSC air-ground dialogue is terminated, normally or due to an abort, the mode is set back to the default value, if it has been set to "dsc".

4.3.5 **The CPDLC-start Service**

4.3.5.1 The *CPDLC-start* service is used to set up a CPDLC dialogue between a CPDLC-air-user and a CPDLC-ground-user. It is a confirmed service, that can be initiated by either the CPDLC-air-user or the CPDLC-ground-user.

4.3.5.2 The SARPs completely determine whether or not a CPDLC-air-user accepts or rejects a CPDLC-ground-users request to establish a CPDLC dialogue.

4.3.5.3 The SARPs place no requirements on whether or not a CPDLC-ground-user accepts or rejects a CPDLC-air-users request to establish a CPDLC dialogue.

4.3.5.4 A CPDLC message can be included in the start request. If the request to open a dialogue is rejected the message is ignored.

4.3.5.5 If a CPDLC message is included in the start request, and the request is accepted, any CPDLC response message required is sent using the CPDLC-message or CPDLC-end services.

4.3.5.6 The CPDLC-start Request/Indication works as follows:

- the initiating ASE
  - receives a CPDLC-start request from its CPDLC-user, and
  - creates an APDU with mode as "cpdlc" and any user supplied message, and

  - invokes the D-START request service primitive providing the APDU, addressing information and QOS parameters, and
  - sets the start timer.
- the receiving ASE
  - receives a D-START Indication, and
  - confirms the validity of the addressing, APDU, and QOS values, and
  - if the ASE is an ground-ASE, confirms that the mode is “cpdlc”, and
  - invokes a CPDLC-start indication to its user containing addressing information, a CPDLC message if provided, and the class of communication, if all of the above checks pass, or
  - aborts if any of the above checks fails.

4.3.5.7 The CPDLC-start Response/Confirmation works as follows:

- the receiving ASE
  - receives a CPDLC-start response from its CPDLC-user, and
  - invokes the D-START service with “accepted” if DSC is “false”, and the *Result* parameter value is “accepted”, and no *Reject Reason* parameter is supplied by the user, or
  - invokes the D-START service with “rejected (permanent)” if DSC is “false”, and the *Result* parameter value is “rejected” and an creates an APDU based on the *Reject Reason* parameter.
- the initiating ASE
  - receives a D-START Confirmation, and
  - invokes a CPDLC-start confirmation to its user indicating that the request to initiate a dialogue was accepted if DSC is “false”, and the *Result* parameter is “accepted”, and there is no *User Data* parameter, or
  - invokes a CPDLC-start confirmation to its user indicating that the request to initiate a dialogue was rejected if DSC is “false”, and the *Result* parameter is “rejected (permanent)” and the source is the DS User, and provides *User Data* parameter as the reason for the rejection of the dialogue request if provided, and
  - stops start timer, if a CPDLC-start confirmation is invoked, or
  - aborts if any of the checks to issue a CPDLC-start confirmation fails.

4.3.6 The DSC-start Service

4.3.6.1 The *DSC-start* service is used to set up a DSC dialogue between a CPDLC-air-user and a CPDLC-ground-user. It is a confirmed service, that can only be initiated by the CPDLC-air-user.

4.3.6.2 The SARPs place no requirements on whether or not a CPDLC-ground-user accepts or rejects a CPDLC-air-users request to establish a DSC dialogue.

4.3.6.3 A CPDLC message can be included in the start request. If the request to open a dialogue is rejected the message is ignored.

4.3.6.4 If a CPDLC message is included in the start request, and the request is accepted, any CPDLC response message required is sent using the CPDLC-message service or DSC-end service.

4.3.6.5 The DSC-start Request/Indication works as follows:

- the initiating CPDLC-air-ASE
  - receives a DSC-start request from its CPDLC-air-user,
  - creates an APDU with mode as "dsc" and any user supplied message,
  - invokes the D-START request service primitive providing the APDU, addressing information and QOS parameters, and
  - sets the start timer.
- the receiving CPDLC-ground-ASE
  - receives a D-START Indication,
  - confirms the validity of the addressing, APDU, and QOS values,
  - if the ASE is an ground-ASE, confirms that the mode is "dsc"
  - invokes a DSC-start indication to its user containing addressing information, a CPDLC message if provided, and the class of communication, and sets DSC to "true", if all of the above checks pass.

4.3.6.6 The DSC-start Response/Confirmation works as follows:

- the receiving CPDLC-ground-ASE
  - receives a DSC-start response from its CPDLC-user,
  - invokes the D-START service with "accepted" if DSC is "true", and the *Result* parameter value is "accepted", and no *Reject Reason* parameter is supplied by the user, or
  - invokes the D-START service with "rejected (permanent)" if DSC is "true", and the *Result* parameter value is "rejected" and an creates an APDU based on the *Reject Reason* parameter
- the initiating CPDLC-air-ASE
  - receives a D-START Confirmation, and
  - invokes a DSC-start confirmation to its user indicating that the request to initiate a dialogue was accepted if DSC is "true", and the *Result* parameter is "accepted", and there is no *User Data* parameter, or
  - invokes a DSC-start confirmation to its user indicating that the request to initiate a dialogue was rejected if DSC is "true", and the *Result* parameter is "rejected (permanent)" and the source is the DS User, and provides *User Data* parameter as the reason for the rejection of the dialogue request if provided, and
  - stops start timer , if a DSC-start confirmation is invoked, or
  - aborts if any of the checks to issue a DSC-start confirmation fails.

4.3.7 **The CPDLC-message Service**

4.3.7.1 The *CPDLC-message* service allows the exchange of air-ground CPDLC messages on CPDLC and DSC dialogues. A CPDLC-message service request can be invoked by either the air or ground user. This service is unconfirmed.

4.3.7.2 The CPDLC-message service Request/Indication works as follows:

- the initiating ASE
  - receives a CPDLC-message request from its user,
  - creates an APDU based on the user supplied message
  - invokes the D-DATA service to pass the APDU
- the receiving ASE
  - receives a D-DATA indication
  - invokes a CPDLC-message indication to its user containing a CPDLC message if the APDU in the D-DATA indication is valid, or
  - aborts if the APDU in the D-DATA indication is not valid

4.3.8 The CPDLC-end Service

4.3.8.1 The CPDLC-end service allows the normal (non-abort) closure of a CPDLC dialogue. It also provides for a transition of an NDA to a CDA, providing an NDA link exists. It is a confirmed service that can only be initiated by the CPDLC-ground-user.

4.3.8.2 The CPDLC SARPs specifically outline requirements that require a CPDLC-air-user to accept or reject the CPDLC-end request.

4.3.8.3 The CPDLC SARPs do not wholly determine the CPDLC-air-user response.

4.3.8.4 The CPDLC-end Request/Indication works as follows:

- the initiating CPDLC-ground-ASE
  - receives a CPDLC-end request from its CPDLC-ground-user, and
  - creates an APDU when a CPDLC message is supplied by its user, if DSC is “false”, and
  - invokes the D-END request service primitive providing the APDU if any.
- the receiving CPDLC-air-ASE
  - receives a D-END Indication,
  - confirms that DSC is “false”, and any APDU included is valid
  - invokes a CPDLC-end indication to its user containing a CPDLC message if provided, if all of the above checks pass, or
  - aborts if any of the above checks fails.

4.3.8.5 The CPDLC-end Response/Confirmation works as follows:

- the receiving CPDLC-air-ASE
  - receives a CPDLC-end response from its CPDLC-air-user,

    - invokes the D-END service with “accepted” if DSC is “false”, and the *Result* parameter value is “accepted”, and includes an APDU based on the user provided message, if any, or
    - invokes the D-END service with “rejected” if DSC is “false”, and the *Result* parameter value is “rejected” and includes an APDU based on the user provided message, if any.
- the initiating CPDLC-ground-ASE
    - receives a D-END Confirmation, and
    - invokes a CPDLC-end confirmation to its user indicating that the request to end a CPDLC dialogue was accepted and includes any *User Data* provided as a *CPDLC Message*, if DSC is “false”, and the *Result* parameter is “accepted”, and when there is a *User Data* parameter it is valid, or
    - invokes a CPDLC-end confirmation to its user indicating that the request to end a CPDLC dialogue was rejected and includes any *User Data* provided as a *CPDLC Message,* if DSC is “false”, and the *Result* parameter is “rejected”, and when there is a *User Data* parameter it is valid, and
    - aborts if any of the checks to issue a CPDLC-End confirmation fails.

4.3.9 The DSC-end Service

4.3.9.1 The DSC-end service allows the normal (non-abort) closure of a DSC dialogue. It is a confirmed service that can only be initiated by the CPDLC-air-user.

4.3.9.2 The CPDLC SARPs specifically outline requirements that require a CPDLC-ground-user to accept or reject the DSC-end request.

4.3.9.3 The CPDLC SARPs do not wholly determine the CPDLC-ground-user response.

4.3.9.4 The DSC-end Request/Indication works as follows:

- the initiating CPDLC-air-ASE
    - receives a DSC-end request from its CPDLC-air-user, and
    - creates an APDU when a CPDLC message is supplied by its user, if DSC is “true”, and
    - invokes the D-END request service primitive providing the APDU if any.
- the receiving CPDLC-ground-ASE
    - receives a D-END Indication,
    - confirms that DSC is “true”, and any APDU included is valid
    - invokes a DSC-end indication to its user containing a CPDLC message if provided, if all of the above checks pass, or
    - aborts if any of the above checks fails.

4.3.9.5 The DSC-end Response/Confirmation works as follows:

- the receiving CPDLC-ground-ASE
    - receives a DSC-end response from its CPDLC-ground-user,
    - invokes the D-END service with “accepted” if DSC is “true”, and the *Result* parameter value is “accepted”, and includes an APDU based on the user provided message, if any, or

- invokes the D-END service with "rejected" if DSC is "true", and the *Result* parameter value is "rejected" and includes an APDU based on the user provided message, if any.

- the initiating CPDLC-air-ASE
  - receives a D-END Confirmation, and
  - invokes a DSC-end confirmation to its user indicating that the request to end a DSC dialogue was accepted and includes any *User Data* provided as a *CPDLC Message*, if DSC is "true", and the *Result* parameter is "accepted", and when there is a *User Data* parameter it is valid, or
  - invokes a DSC-end confirmation to its user indicating that the request to end a DSC dialogue was rejected and includes any *User Data* provided as a *CPDLC Message,* if DSC is "true", and the *Result* parameter is "rejected", and when there is a *User Data* parameter it is valid, and
  - aborts if any of the checks to issue a DSC-end confirmation fails.

4.3.10 The CPDLC-forward Service

4.3.10.1 The CPDLC-forward service allows the CPDLC-ground-user to send a CPDLC message to another CPDLC-ground-user. This service is confirmed.

4.3.10.2 The CPDLC-forward service is "one-shot". Thus the receiving CPDLC-ground-ASE always rejects the dialogue start request. There is no CPDLC-ground-user forward-response.

4.3.10.3 The CPDLC-forward Request/Indication works as follows:

- the initiating CPDLC-ground-ASE
  - receives a CPDLC-forward request from its CPDLC-ground-user, and
  - creates an APDU based on the user supplied message, and
  - invokes the D-START request service primitive providing the APDU, addressing information and QOS parameters, and
  - sets the start timer.
- the receiving CPDLC-ground-ASE supports the forward service
  - receives a D-START Indication, and
  - determines version number is compatible, and
    - confirms the validity of the addressing, APDU, the QOS values, and if OK:
      - invokes a CPDLC-forward indication to its user containing addressing information, and the CPDLC message provided, and
      - creates an APDU indicating success, and
      - invokes a D-START response with the APDU, and rejects the dialogue request, or
      - aborts if any of the above checks fails, or
  - determines version number is incompatible, and
    - creates an APDU indicating version incompatibility, and
    - invokes a D-START response with the APDU, and rejects the dialogue request, or
- the receiving CPDLC-ground-ASE does not support the forward service

  - receives a D-START Indication, and
    - creates an APDU indicating service not supported, and
    - invokes a D-START response with the APDU, and rejects the dialogue request

4.3.10.4 The CPDLC-forward Confirmation works as follows:

- the initiating ASE
  - receives a D-START Confirmation, and
  - invokes a CPDLC-start confirmation to its user if the version number is compatible, and if the *Result* parameter is "rejected(permanent)", and the *User Data* parameter is valid, or
  - invokes a CPDLC-start confirmation to its user providing the APDU and the receiving CPDLC-ground-ASE version number, if the version number is not compatible, and if the *Result* parameter is "rejected(permanent)", and the *User Data* parameter is valid, or
  - stops start timer , if a CPDLC-forward confirmation is invoked, or
  - aborts if any of the checks to issue a CPDLC-forward confirmation fails.

4.3.11 The CPDLC-user-abort Service

4.3.11.1 The CPDLC-user can abort a CPDLC or DSC dialogue at any time, subject to operational or procedural reasons. The CPDLC SARPs section 2.3.7 provide several reasons when the CPDLC-user must invoke a CPDLC or DSC dialogue. These reasons are considered operationally catastrophic. When a CPDLC-user invokes a CPDLC-user-abort the user can supply a reason. Messages in transit may be lost during this operation. This is an unconfirmed service.

4.3.11.2 The CPDLC-user-abort service can be invoked at any time that the CPDLC-user is aware that any CPDLC or DSC service is in operation. After this primitive is invoked, no further primitives may be invoked for the current dialogue.

4.3.11.3 The CPDLC-user-abort Request/Indication works as follows:

- the initiating ASE
  - receives a CPDLC-user-abort request from its user, and
  - stops any timer that is set, and
  - creates an APDU based on the supplied CPDLC message when provided, or
  - else creates an APDU with the element "undefined", and
  - invokes D-ABORT with the APDU indicating that the originator of the abort is a user of the ASE, and
  - sets DSC to "false".
- the receiving ASE
  - receives a D-ABORT indication, and
  - determines that the originator is user and the APDU is valid, and its user is active, and
  - stops any timer, and
  - sets DSC to "false" if DSC is "true".

4.3.12 The CPDLC-Provider Abort Service

4.3.12.1 When the CPDLC-ASE detects an error from which it cannot recover, the CPDLC-provider-abort service is invoked when the ASE is able to do so. This is a CPDLC provider-initiated service with a single, mandatory parameter telling why the dialogue is being aborted. Messages in transit may be lost during this operation.

4.3.12.2 The CPDLC-provider-abort indication works as follows;

- the initiating ASE
  - detects an error or fails, and
  - if able stops any timer, and
  - if able, creates an APDU with the Reason “undefined-error”, and
  - if able, provides a CPDLC-provider-indication to its CPDLC user containing the APDU, if the user is active, and
  - if able, invokes D-ABORT with user as the originator and the APDU, and
  - if able, sets DSC to “false”, if DSC is “true”.

- the receiving ASE
  - receives a D-ABORT indication and determines that the originator is provider and the data is valid and issues a CPDLC-provider-abort containing the provider reason, to its user, if the user is active, or
  - receives a D-P-ABORT indication and the data is valid and issues a CPDLC-provider-abort containing the provider reason, to its user, if the user is active, or
  - stops any timer, and
  - sets DSC to “false”, if DSC is “true”.

4.4 CPDLC SARPs Section Description

4.4.1 CPDLC SARPs Section 2.3.2: General Requirements

4.4.1.1 *Version Number*

4.4.1.1.1 The CPDLC SARPs section 2.3.2.1 is included to allow the Context Management (CM) application to exchange version numbers of the CPDLC application. It is necessary to allow for future versions of the protocol to be negotiated by CM. It has no effect on the CPDLC functionality.

4.4.1.2 *Error Processing Requirements*

4.4.1.2.1 In the abstract service definition, each service has a set of parameters and the abstract syntax of those parameters specified. Thus information which is not a valid syntax is not allowed to be input.

4.4.1.2.2 In the protocol definition, there is a requirement that no service is permitted to be called when an ASE is in an incompatible state. Thus making use of the abstract services is not permitted at these times.

4.4.1.2.3 An implementation should not allow the user to take invalid actions; however, there is no requirement to prevent an implementation from allowing this. The error processing requirements section thus says that *if* the implementation allows the user to enter invalid information, the system must inform the user that an entry error has occurred. In that case, the error is locally detected and the dialogue does not need to be aborted.

4.4.2 CPDLC SARPs Section 2.3.3: The Abstract Service

4.4.2.1 *The Concept of an Abstract Service*

4.4.2.1.1 The CPDLC SARPs Section 2.3.3 concerns the CPDLC abstract service. The following paragraphs provide an explanation of "Abstract Service".

4.4.2.1.2 In order to define the CPDLC-ASE (i.e. the part of the abstract service provider that contains the protocol machine — see CPDLC SARPs section 2.3.5, it is necessary to describe its reactions to both PDUs arriving from the peer ASE and action from the local CPDLC-user. The PDUs are well defined in the protocol. The actions of the user, however, are not. The SARPs do not attempt to dictate the actions of the user except where absolutely necessary. Despite this, in order to define the ASE it is necessary to have a clear definition of user actions.

4.4.2.1.3 In order to get around this conundrum, an "application abstract service" is defined. The CPDLC abstract service is a technical description of the interactions between the CPDLC-user and the ASE. These interactions are precisely defined in SARPs section 2.3.3. Having this definition allows the ASE to be specified precisely in terms of its reactions to the arrival of PDUs and the invocation of the service primitives by the user. This, therefore, is the reason for defining the abstract service.

4.4.2.1.4 The abstract service is defined as being the description of the interface between the ASE and the ASE service-user. These are known as the CPDLC-ASE and the CPDLC-user. The CPDLC user is generally not the human user; it is that part of the system that uses the CPDLC-ASE.

4.4.2.2 *The Concept of APIs*

4.4.2.2.1 If one was to buy a CPDLC application, one would be buying a suite of executable code. From the code itself it is impossible to know whether or not the abstract service has actually been implemented. Therefore the CPDLC SARPs do not require that the abstract service interface has to be built as a real interface. It only requires that, when one examines it from an external point of view, it behaves in the same way as if it had been built. This is the explanation of the SARPs requirement 2.3.3.1.1.

4.4.2.2.2 Thus the implementors may choose to build a CPDLC application with an Application Programmatic Interface (API) that corresponds to the CPDLC application abstract service, or they may choose not to — it is entirely up to them. However, it should be realized that there are a number of good reasons why one might not want to build a system with an API exactly like the abstract service. Examples include:

- there may be a more efficient way of building the software.
- the abstract service does not include parameters that are needed locally, but do not effect the state machine; for example, an API might include an indication of which aircraft system a CPDLC message has come from.
- it may not be easy to build the abstract service from the development tools that are used being used.
- the abstract service does not have any programming language bindings. An API would require an interface defined in a particular programming language.

4.4.2.2.3 Implementation of the abstract service interface is not mandated by the CPDLC SARPs. The requirements for CPDLC set out by the SARPs are limited in scope — they are designed only to ensure interoperability between CPDLC air and ground systems, or between two CPDLC ground systems, and to ensure that they meet the stated functionality requirements. The CPDLC SARPs do not specify the nature of any internal interface within the software, nor do they specify the human interface. Individual implementation projects should define their own internal interfaces to suit their own requirements.

4.4.2.2.4 In summary, an application abstract service is defined in the SARPs in order to be able to define the ASE protocol machine. It does not have to be built as an API in any implementation, and there are several good reasons why it should not be implemented exactly as defined. A real implementation of the CPDLC SARPs would normally be expected to define its own APIs.

4.4.2.2.5 The CPDLC application abstract service consists of eight functions listed in the CPDLC SARPs section 2.3.3.2.1.

4.4.2.3 *The CPDLC Functional Model*

4.4.2.3.1 Figure 4.4-1 below shows an abstract model for the CPDLC application. Just as with the abstract service, this model shows a design of the CPDLC application, breaking it down into modules. However, there is no requirement that an implementation actually builds it this way. The figure is presented here in order to explain the terms that are used throughout the chapter. It is not required that the design of an implementation follows this structure.

Figure 4.4-1. CPDLC Functional Model

4.4.2.3.2 The figure shows three modules:

- The CPDLC-user (which could be a CPDLC-air-user or a CPDLC-ground-user),
- The control function, and
- The CPDLC-ASE (application service element — which could be a CPDLC-air-ASE or a CPDLC-ground-ASE).

4.4.2.3.3 In addition, it defines the CPDLC application entity as the control function together with the CPDLC ASE.

4.4.2.3.4 Abstract interfaces are shown between the different modules:

- The CPDLC application entity service interface — which is the same as the abstract service interface defined in SARPs section 2.3.3,
- The CPDLC application service element service interface — which is also the same as the abstract service interface defined in SARPs section 2.3.3,

- The dialogue service interface — which is defined in [2], and is identical for all air-ground applications.

4.4.2.3.5 Since the CPDLC application entity service interface is identical with the CPDLC application service element service interface, the control function module passes primitive calls directly from one to the other without interference.

4.4.2.4 *Conventions*

4.4.2.4.1 **Service Primitives**

4.4.2.4.1.1 The CPDLC SARPs define eight CPDLC services. A CPDLC service consists of one to four primitives. Each primitive consists of the name of the CPDLC service (CPDLC-start, DSC-start, CPDLC-message, CPDLC-end, DSC-end, CPDLC-forward, CPDLC-user-abort, and CPDLC-provider-abort) and a suffix that indicates at what point in the service the primitive occurs (request, indication, response, confirmation), e.g. CPDLC-start service request. The primitives are further explained below:

- the CPDLC-user that initiates the service calls on its CPDLC-ASE to perform an action — this is called the "request",

- after the request is passed to the CPDLC-ASE on the other side of the communication link, that ASE uses the service to pass the information on to its CPDLC-user — this is called the "indication",

- the CPDLC-user that receives the indication may choose to respond to it, in which case it calls upon its CPDLC-ASE to send a reply — this is called the "response",

- finally, the CPDLC-ASE receiving the response provides its CPDLC-user (which started the sequence of events) with the information — this is called the "confirmation".

4.4.2.4.1.2 The terms "request", "indication", "response" and "confirmation" come from the field of communication protocols. A given service need not use all four primitives. Some CPDLC services make use of only one (indication), some two (request and indication, or request and confirmation), some three (request, indication and confirmation) and some all four (request, indication, response and confirmation).

4.4.2.4.1.3 The CPDLC-start, DSC-start, CPDLC-end, and DSC-end service use all four service primitives. These primitives are illustrated generically in Figure 4.4-2.

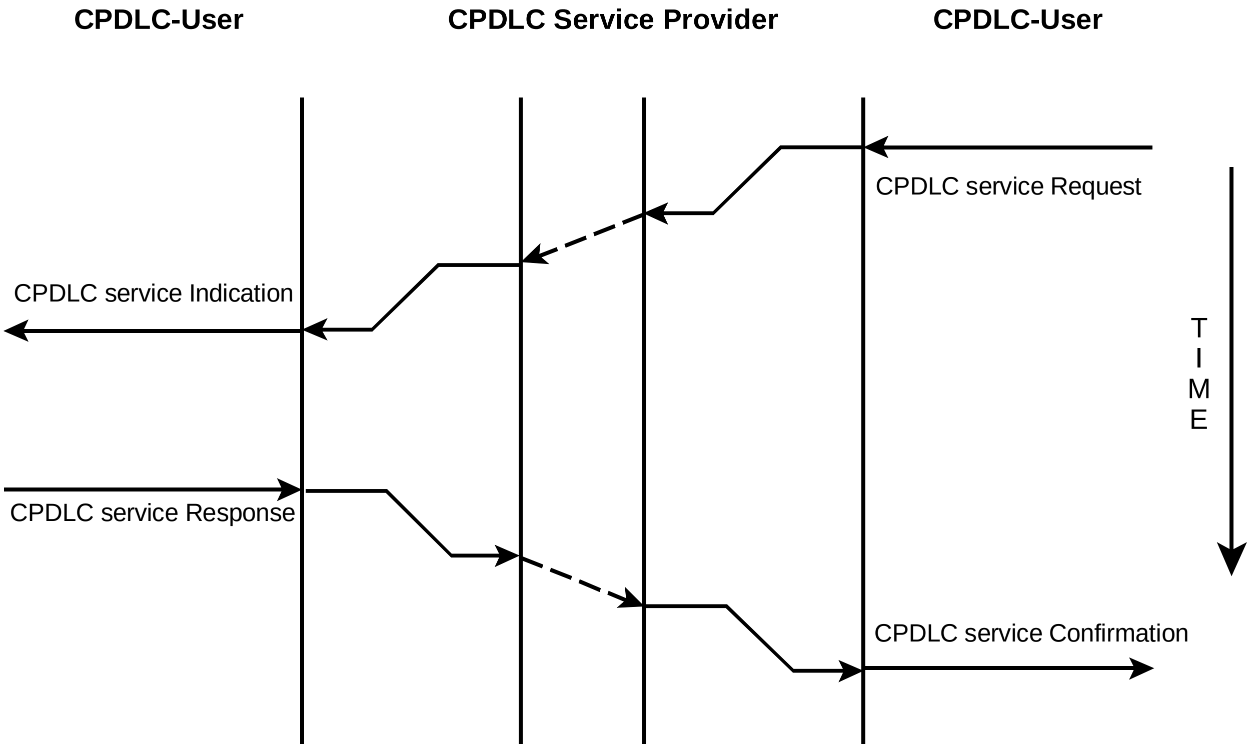


Figure 4.4-2. Generic CPDLC Service Using All Four Service Primitives

4.4.2.4.1.4 When the CPDLC-forward service is supported by the receiving CPDLC-ground-user and the sending and receiving ground systems' versions are equal, the CPDLC-forward service uses three service primitives as illustrated generically in Figure 4.4-3.

4.4.2.4.1.5 The CPDLC services can use only two primitives in two ways: either as a request and indication or as a request and confirmation. When the CPDLC-forward service is not supported by the receiving CPDLC-ground-user or if the sending and receiving ground systems' versions are not equal, the CPDLC-forward service uses only the request and confirmation primitives as illustrated generically in Figure 4.4-4.

4.4.2.4.1.6 The CPDLC-message and CPDLC-user-abort services always use only two service primitives: the request and indication primitives as illustrated generically in Figure 4.4-5.

4.4.2.4.1.7 The CPDLC-provider-abort service always uses only one service primitive (indication) as illustrated generically in Figure 4.4-6. This service is provider initiated and an indication primitive given to both CPDLC-users (if active). The CPDLC-provider-abort results when either a CPDLC-ASE (sender or receiver) or the underlying communication service aborts.

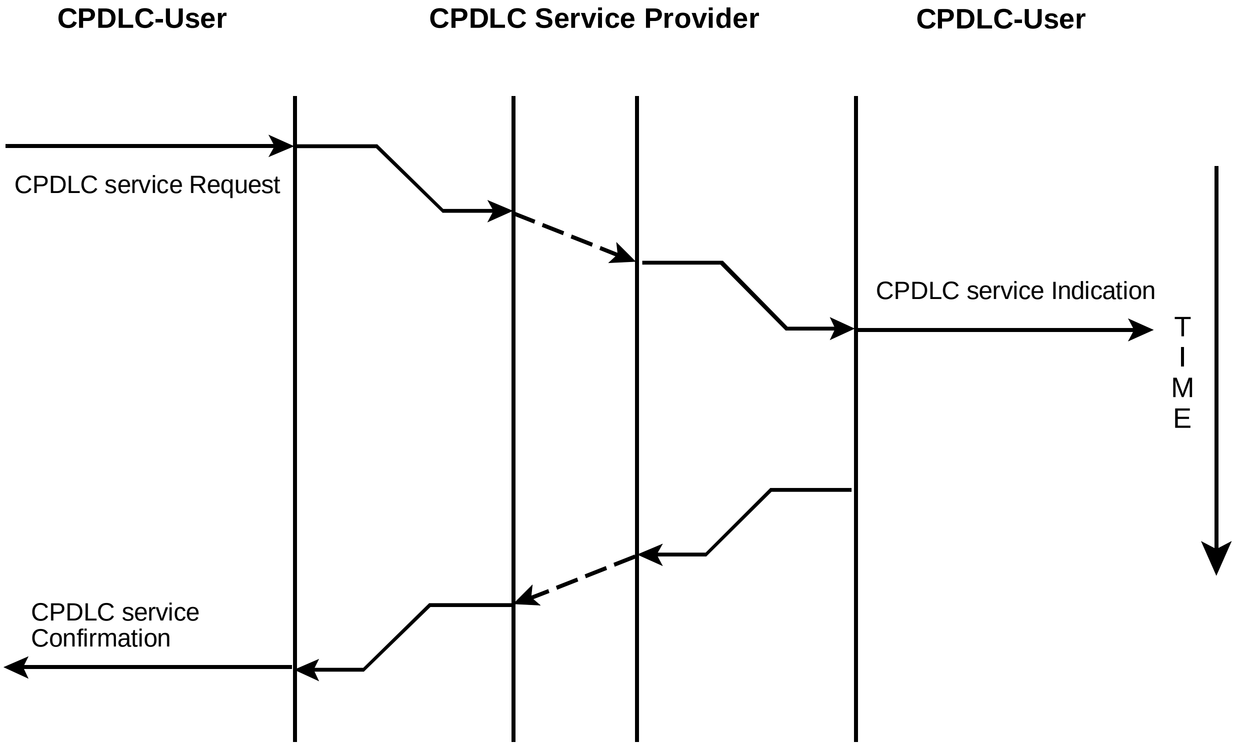


Figure 4.4-3. Generic CPDLC Service Primitives When Three Primitives Are Used

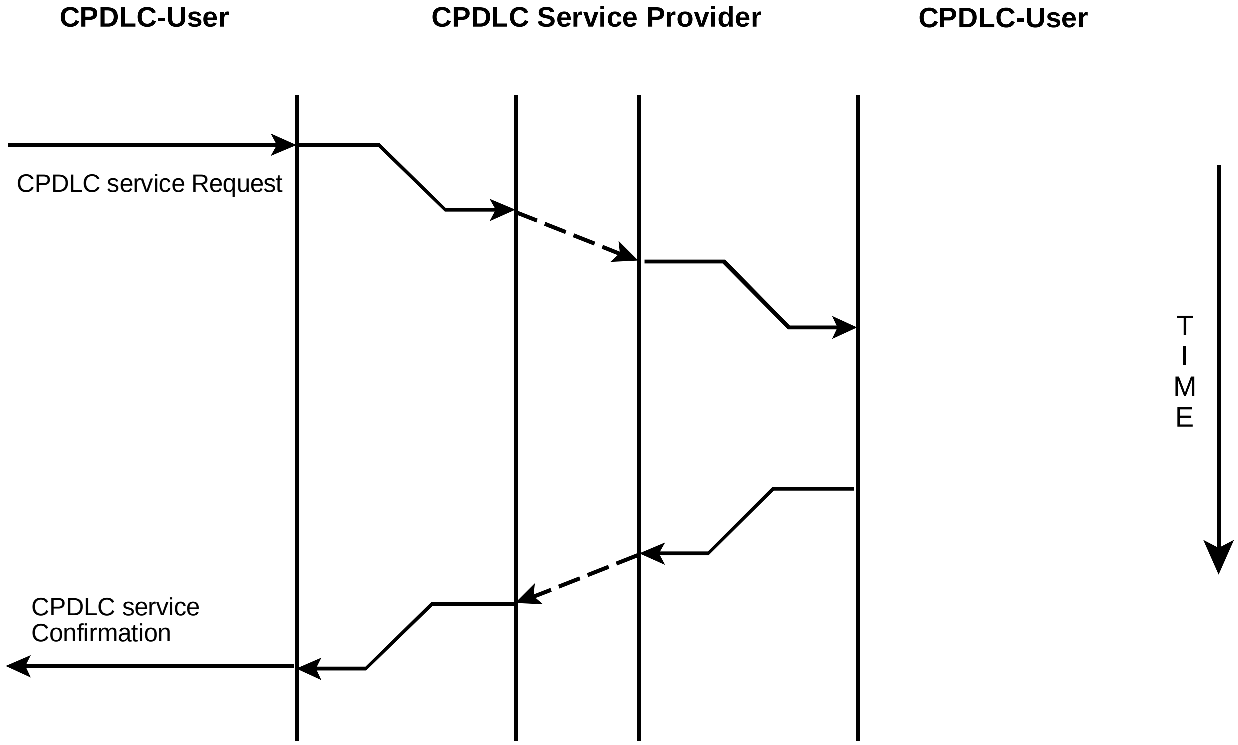


Figure 4.4-4. Generic CPDLC Service Primitives When Two (Request and Confirmation) Primitives Are Used

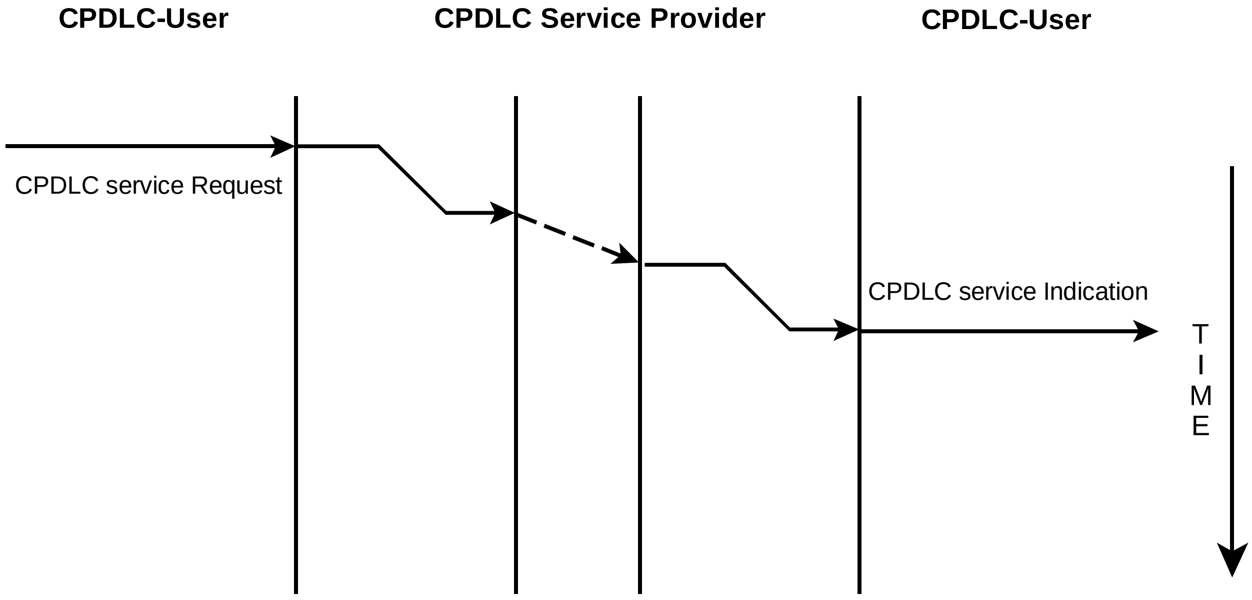


Figure 4.4-5. Generic CPDLC Service Primitives When Two (Request and Confirmation) Primitives Are Used

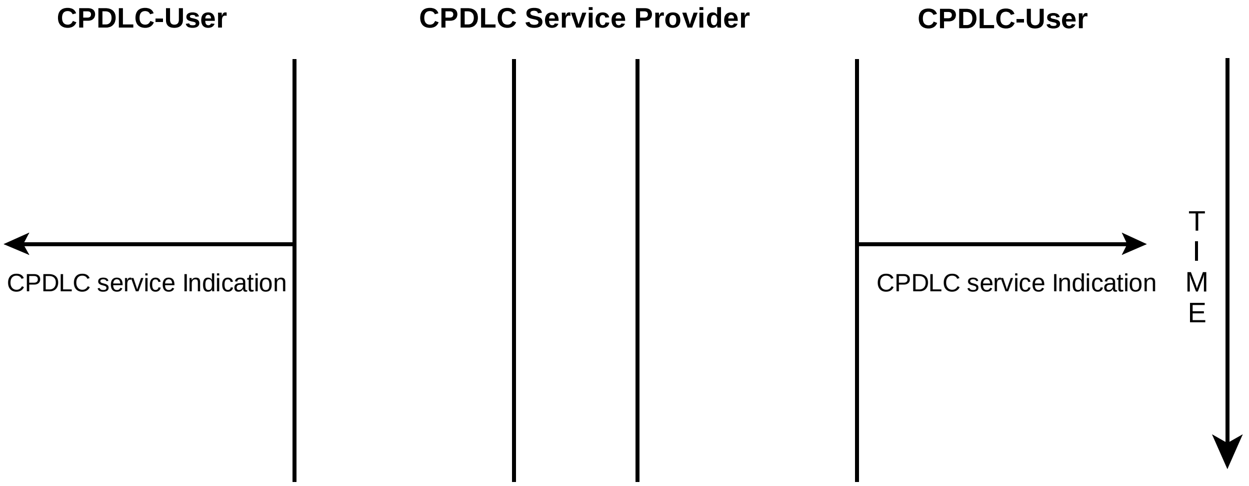


Figure 4.4-6. Generic CPDLC Service Primitive When One (Indication) Primitive Is Used

4.4.2.4.2 Confirmed/Unconfirmed

4.4.2.4.2.1 A *confirmed CPDLC service* is one that involves a handshake between the user that requests the service and the user or ASE that is informed that the service has been requested. Figure 4.4-, Figure 4.4-3, and Figure 4.4-4 illustrate the CPDLC confirmed services.

4.4.2.4.2.2 An *unconfirmed CPDLC service* involves no handshake. Figure 4.4-5 illustrates the CPDLC unconfirmed services.

4.4.2.4.2.3 The CPDLC-provider-abort service is neither confirmed or unconfirmed, since it does not involve CPDLC-user initiation.

4.4.2.4.3 **Detailed Service Descriptions**

4.4.2.4.3.1 **Introduction**

4.4.2.4.3.1.1 Each of the detailed service descriptions is defined in the same way. First, there is either one or two tables indicating the parameters of the service and their status in each of the primitives. Second, each parameter has a short description, including a description of its abstract syntax. This is further described in the following sections.

4.4.2.4.3.2 **Service Primitives and Parameters**

4.4.2.4.3.2.1 Each CPDLC service has a set of parameters. The set of parameters used in the request, indication, response, and confirmation primitives may be different. For example, the CPDLC-start request primitive has the *Called Peer Identifier*, the *Calling Peer Identifier*, the *CPDLC Message*, and the *Class of Communication* parameters, while the CPDLC-start response primitive has the *Reject Reason* and *Result* parameters. If a parameter is not defined for a given primitive it cannot be used in that primitive.

4.4.2.4.3.3 The services are depicted in the CPDLC SARPs by primitive and parameter tables as shown in Table 4.4-1. Not all CPDLC services require each primitive to be used. That is, if indication of a particular primitive is not needed due to reasons like redundant information being relayed, then the parameter column for that primitive is omitted. If a parameter column for a primitive is present, but all of the parameters are left blank, this means that the primitive is used by the CPDLC service, but does not contain any data.

Table 4.4-1. Description of the CPDLC Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
CPDLC Parameter 1	M			
CPDLC Parameter 2	M	M(=)		
CPDLC Parameter 3	U	C(=)		
CPDLC Parameter 4	C	C(=)		
CPDLC Parameter 5				C
CPDLC Parameter 6	U	M		C
CPDLC Parameter 7				M

4.4.2.4.3.4 For a specific primitive, each parameter is described by a value that dictates the terms under which that parameter is used. If the use of any parameter does not follow the rules as set forth by the primitive and parameter tables, there is an error in the implementation. The abbreviations used in the primitive and parameter sections are described below:

- blank — this means that the specific parameter will not be used in this service primitive.

- C — this means that the parameter is conditional upon some state. A “C” differs from a “U” (User Option) due to the fact that if the stated condition exists, the parameter must be supplied, while a “U” means that the parameter's use is wholly up to the user. The specific conditions for a “C” are contained in the CPDLC SARPs section 2.3.3.

- C(=) — this means that a parameter is conditional upon the value of the parameter to the left of it in the table being present, and if it is present, the “C(=)” parameter will retain the parameter to the left of it in the tables. In CPDLC a “C(=)” is typically used when a parameter request or response has an optional or conditional parameter to its left.

- M — this means that the parameter is always present, and no option not to use it exists.

- M(=) — this means that the parameter is always present, and will always take the value to the left of it in the table; and no option not to use it exists.

- U — this means that the use of the parameter is a user option. Therefore the presence of the parameter is optional, and will be used based upon user requirements. Some of the user requirements for "U" parameters are specified in the CPDLC SARPs section 2.3.7. The use of "U" parameters is not wholly specified in the SARPs. Further requirements determining the use of "U" parameters may be determined by operational or procedural requirements outside of the scope of the CPDLC SARPs.

4.4.2.4.3.5 The CPDLC-forward service is presented with two parameter tables. The reason for this is that this service may operate in one of two ways. The CPDLC-forward service may operate with only the request, indication, and confirmation parameters; or may operate with only the request and confirmation parameters.

4.4.2.4.4 Service Parameters

4.4.2.4.4.1 Throughout the service descriptions every parameter is described. In particular, there is a note that explains the purpose of the parameter, and a requirement that states what values it may contain.

4.4.2.4.4.2 In many cases, the primitive parameters have the same contents than APDU fields in the ASN.1 description. The CPDLC-ASE is tasked to copy the parameter value within the APDU field without modification. In order to avoid confusion by defining identical data structures twice, the type of those primitive parameters is specified by simply referring to the corresponding ASN.1 type in the APDU. The ASN.1 is used in the service definition as a syntax notation only and does not implicitly imply any local encoding of these parameters. The implementation of these parameters remains a local implementation issue.

4.4.2.4.4.3 For other parameters, the syntax is described by enumerating the authorized abstract values.

4.4.2.4.5 CPDLC Services

4.4.2.4.5.1 This section lists the CPDLC services, and explains why each is a confirmed or an unconfirmed service.

4.4.2.4.5.2 The CPDLC-start and DSC-start are confirmed services. The peer CPDLC-user must respond to the start services indicating acceptance or rejection of the request to establish a dialogue. This information and any user supplied rejection reason is confirmed to the initiating user.

4.4.2.4.5.3 The CPDLC-end and DSC-end are confirmed services. The peer CPDLC-user must respond to the end services indicating acceptance or rejection of the request to end a dialogue. This information and any user supplied data is confirmed to the initiating user.

4.4.2.4.5.4 The CPDLC-message service is an unconfirmed service. Requirements for receipt of a CPDLC message are governed by operational requirements rather than technical communication service requirements. A CPDLC message interaction may involve a series of required, and not necessarily paired CPDLC message exchanges. For example a given message may require both a logical and at least one operational indication of message receipt and processing. The burden of CPDLC message response is therefore left totally on the CPDLC-user. No requirement is placed on a receiving ASE using the message service to get a response from its user, nor respond itself. Thus the CPDLC message service is unconfirmed.

4.4.2.4.5.5 The CPDLC-forward service is a confirmed service. The peer CPDLC-ASE must respond to the forward service indicating rejection of the request to establish a dialogue and the status of the forward function. This information is confirmed to the initiating user.

4.4.2.4.5.6 The CPDLC-user abort service is an unconfirmed service since there is no requirement for the initiating user to know that an abort has arrived at the peer.

4.4.2.4.5.7 The concept of confirmed or unconfirmed does not apply to the CPDLC-provider-abort service since it is never CPDLC-user initiated.

4.4.3 CPDLC SARPs Section 2.3.4: Formal Definition of Messages

4.4.3.1 *Introduction*

4.4.3.1.1 CPDLC section 2.3.4 abstract syntax is written in a notation that is called ASN.1. It is strongly recommended one is familiar with ASN.1 in order to understand the details of section 2.3.4.

4.4.3.2 *CPDLC SARPs 2.3.4 Encoding/Decoding Rules*

4.4.3.2.1 This section defines which APDU the CPDLC systems must be able to encode and decode.

4.4.3.2.2 An APDU (Application Protocol Data Unit) is a sequence of bits that are passed from one peer application to another. The sequence of bits is a concrete representation of a message that is passed between CPDLC applications. The ASN.1 defines all CPDLC messages that are exchanged between CPDLC-users. An APDU for this message is a sequence of bits representing the information that is exchanged.

4.4.3.2.3 A CPDLC system is not required to encode or decode all messages specified in the ASN.1 description. A CPDLC-air system is not required to be able to encode a GroundPDUs nor to decode an AircraftPDUs. A CPDLC-ground system is not required to be able to encode an AircraftPDUs but must be able to encode and decode a GroundPDUs.

4.4.3.3 *CPDLC ASN.1 Abstract Syntax*

4.4.3.3.1 This section defines the abstract syntax of the protocol. That is, it defines the structure of the PDUs that are to be sent between a aircraft and a ground system or between two ground systems.

4.4.3.3.2 Data types exchanged by CPDLC-ASEs are described in the CPDLC SARPs by using a machine-independent and language-independent syntax. There is no constraint put on the implementors concerning the machine nor the development language to be selected for implementing the protocol.

4.4.3.3.3 The ASN.1 module *CPDLCMessageSetVersion1* contains the data types of the protocol data units handled by the CPDLC-ASEs. Unlike common Open Systems Interconnection (OSI) ASEs (e.g. Application Control Service Element (ACSE)), no object identifier has been attached to the CPDLC ASN.1 specification. Indeed, the Upper Layers Communications Service (ULCS) architecture releases the applications from negotiating, during the dialogue establishment, the applicable abstract syntax. Object identifiers related to the CPDLC application (application context name and version number) are defined in [2].

4.4.3.3.4 The CPDLC ASN.1 is organized as follows:

- the type of APDUs
- the message header,
- the message element choices,
- the message element variables definitions in alphabetical order

4.4.3.3.5 A hierarchical description of the ASN.1 message variables and definitions of the variables of given in section 6 of this chapter.

4.4.3.3.6 **ASN.1 Tags**

4.4.3.3.6.1 Tags are used in ASN.1 to allow to distinguish data types when confusion is possible. For instance, when a data type contains two optional elements of the same type, if only one is encoded there is no means for the decoder to know which element the decoded value is attached.

4.4.3.3.6.2 Even if tag values are not used by the Packed Encoding Rules, the ASN.1 grammar mandate the use of tags in some cases. When specifying the CPDLC data types, the following rules have been used:

- tags are always used within CHOICE data type, starting at 0 and then incremented by 1 for each entry.

- tags are not used at all in SEQUENCE data type when no confusion is possible. When any optional element is defined, all elements in the sequence are tagged.

4.4.3.3.7 **Extensibility Markers**

4.4.3.3.7.1 In order to allow the upgrade of the ASN.1 specification when new CPDLC capabilities or message elements are made available, the extensibility ASN.1 feature (ellipse) has been used in some data types in the CPDLC service. For example, extensions have been allowed for the following:

- the type of Ground and Aircraft PDUs,
- user or provider abort reasons,
- message element choices,
- the type of clearance provided,
- error reasons,
- the type of speed, and
- the type of traffic.

4.4.3.3.8 **Handling Numerical Values**

4.4.3.3.8.1 Some operational data are real. REAL data types exist in ASN.1 but the associated encoding procedures are not optimized in term of data size (Packed Encoding Rules (PER) do not specify specific rules for real but import the Basic Encoding Rules (BER) ones which requires the encoding of a mantissa, a base and a exponent taking several octets). Real values are therefore specified using the INTEGER data type.

4.4.3.3.8.2 ASN.1 constraints are attached to the INTEGER data type to take advantage of both range and resolution. The lower bound is equal to the operational minimal value devised by the resolution and the upper bound is equal to the operational maximal value devised by the resolution.

4.4.3.3.8.3 For instance:

SpeedMach ::= INTEGER (500..4000)

-- unit = Mach Range (0.5 to 4.0), resolution = 0.001

4.4.3.3.8.4 The definition of real fields using the INTEGER data type, allows for a very simple algorithm for the computing operational value from the encoded value and vice-versa. The message sender divides the operational value by the resolution to find the value to send. The receiver inverses the procedure, and multiplies the received value by the resolution to get the operational value.

4.4.3.3.9 **Entry Points**

4.4.3.3.9.1 The top level (which will typically be used as entry point in any ASN.1 compiler) consists of two main structures: the AircraftPDUs is a choice of time-stamped PDUs generated by the aircraft, and GroundPDUs is a choice of time-stamped PDUs generated by the ground system.

4.4.3.3.10 **Time Representation**

4.4.3.3.10.1 Data types have been specified for containing time indication (Date, DateTimeGroup, Year, Month, Hours, Minutes, and Seconds). This way of representing time is preferred over the pre-defined ASN.1 representations (GeneralizedTime and UTCTime) for optimization of the PER encoding.

4.4.4 CPDLC SARPs Section 2.3.5: Protocol Definition

4.4.4.1 *Message Sequence Diagrams*

4.4.4.1.1 Time sequence diagrams or message sequence diagrams are used to denote the relationship between the primitives that form a CPDLC service and the order in which they occur., e.g. the indication/confirmation primitives occurs some time after the request/response primitives.

4.4.4.1.2 Inherent to the service model is the notion of queuing, as illustrated in Figure 4.4-7. The CPDLC-service indications and confirmations are delivered to the CPDLC-users in the order that the corresponding CPDLC-service requests and responses were issued. One exception to the notion of queuing is the abort services (CPDLC-user-abort and CPDLC-provider-abort services) which may overtake other primitives and empty the primitives in the queue.

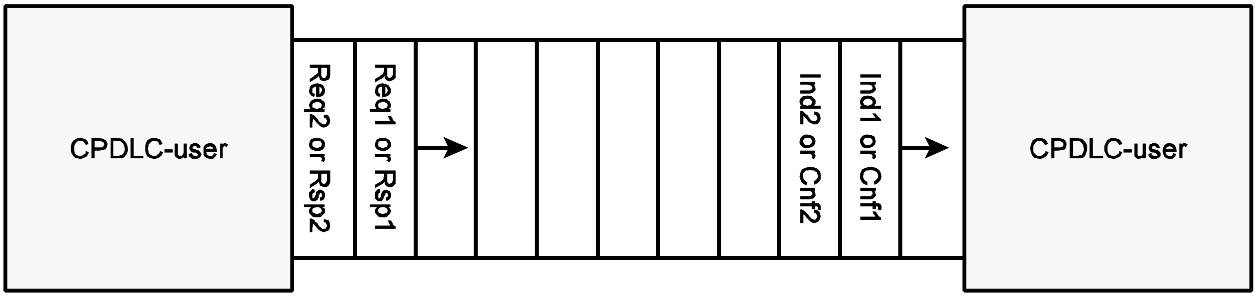


Figure 4.4-7. Communication Queue

4.4.4.1.3 Each figure in the CPDLC SARPs section 2.3.5 has the same structure as illustrated in Figure 4.4-8. There are four vertical lines that separate the five major components in the CPDLC system. From left to right, they are:

- The CPDLC-ground-user — that part of the ground system that processes the CPDLC information to provide or receive information to/from human users,
- The CPDLC-ground-ASE — that part of the ground system that implements the CPDLC protocol,
- The dialogue service provider — that part of the ground system, the air system and the networks that, together, provide the dialogue service, as defined in the upper layer architecture — on the figures this is the thin strip down the middle,
- The CPDLC-air-ASE — that part of the avionics system that implement the CPDLC protocol, and
- The CPDLC-air-user — that part of the avionics system that processes the CPDLC information to provide or receive information to/from human users.

4.4.4.1.4 The middle three sections of the diagrams (CPDLC-ground-ASE, dialogue service provider and CPDLC-air-ASE) together form the CPDLC service provider, and are labeled as such on the diagrams.

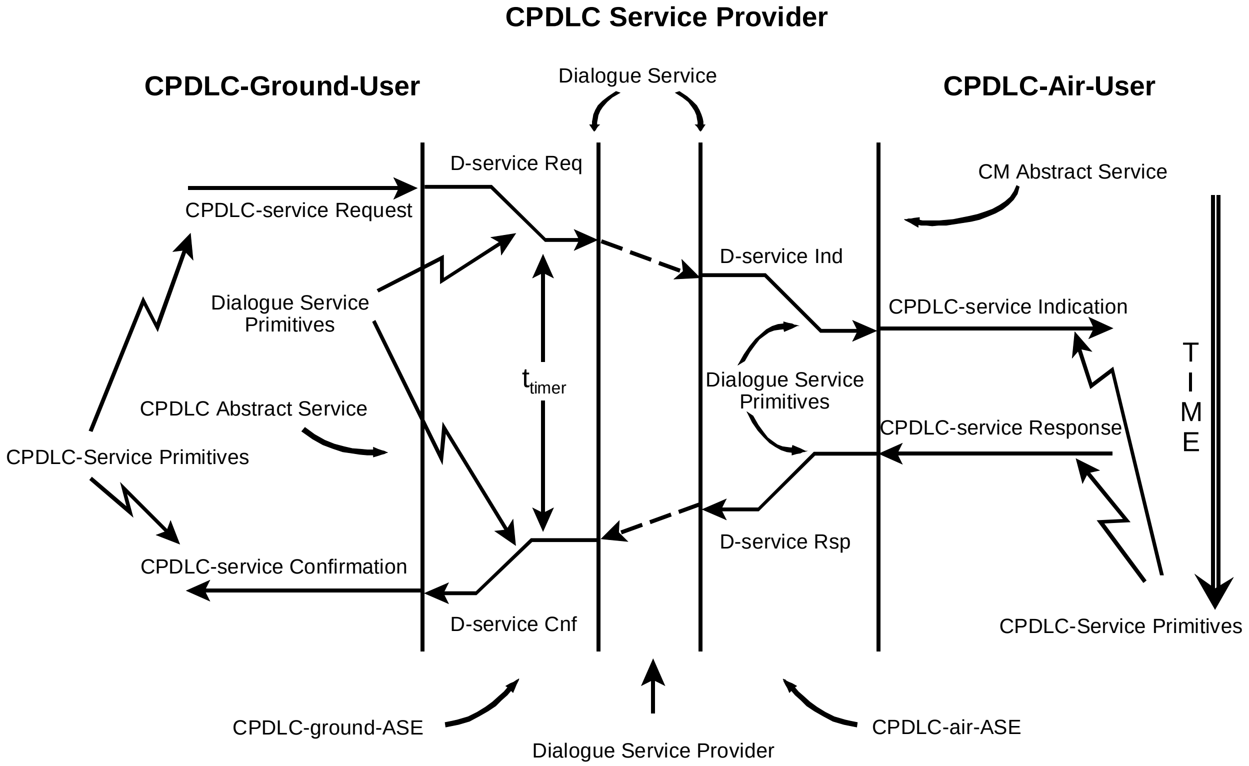


Figure 4.4-8. Message Sequence Diagram

4.4.4.1.5 The outer two vertical lines represent the CPDLC abstract service. Any lines crossing them represent the invocation of one of the CPDLC service primitives. The CPDLC service primitives are labeled in the CPDLC user part of the figure.

4.4.4.1.6 The inner two vertical lines represent the Dialogue service. Any lines crossing them represent the invocation of one of the Dialogue service primitives. The Dialogue service primitives are labeled in the CPDLC-ASE part of the figure.

4.4.4.1.7 The diagrams represent a sequence of events. Time is always considered to run down the figure from the top (representing the earliest time) to the bottom (representing the latest time).

4.4.4.1.8 If the ASEs set timers, they are marked on the figures by vertical lines with arrows at both ends. This is depicted in figure 4.4-7 by the "t_{timer}" label.

4.4.4.1.9 It should be noted that the last figures representing an abort situation can be overlaid on top of any of the other figures to represent an abort in action.

4.4.4.2 *CPDLC SARPs 2.3.5.2 CPDLC Service Provider Timers*

4.4.4.2.1 This section lists the service timers that are defined in the protocol, and suggests values for them.

4.4.4.2.2 The purpose of the service provider timers in the CPDLC service is not operational. For operational reasons, there may be a requirement to have other timers that are shorter than the timers described here. The purpose of the service provider timers is only to ensure that the ASE protects itself when communicating with a system that fails to respond.

4.4.4.2.3 For CPDLC, almost all CPDLC information sent over a CPDLC or DSC dialogue involves human input. For this reason the use of service provider timers occurs only in the CPDLC-start, DSC-start, and CPDLC-forward services. Since a CPDLC-user is prohibited from initiating any other action on a given connection until the connection is fully established, service provider timers have been provided to minimize the "lock-out" time. It should also be noted that a user is not permitted to include any operational response when accepting the request to establish a CPDLC or DSC dialogue, to minimize the operational time required.

4.4.4.2.4 Most, if not all, operational system will require operational timers of much shorter duration than the values set for the service provider timers. The operational timers can vary according to differing operational environments. The CPDLC SARPs to not specify the value of operational timers, nor require their implementation.

4.4.4.2.5 The assignment of values for timers must be optimized based on operational testing of the application. In such testing, incompatible timer values and optimum combinations can be identified. Implementations of CPDLC protocol are required to support configurable values for all timers and protocol parameters, rather than having fixed values. This allows modification as operational experience is gained.

4.4.4.3 CPDLC SARPs Sections 2.3.5.3 Through 2.3.5.6 CPDLC-ASE Protocol Description

4.4.4.3.1 The CPDLC service provider is described in the SARPs as finite state machines or protocol machines (PM). The protocol machine for a particular service starts in an initial state (IDLE). Events, which are service primitives received from the CPDLC-users above or the Dialogue Service provider below, as they occur, trigger activity on the part of the PM. As part of this activity, actions may be required (service primitives issued to the CPDLC-users and/or the underlying Dialogue Service provider).

4.4.4.3.2 The protocol description explains the rules by which the ASEs work. There is a detailed specification of actions taken by the ASEs when triggered by certain events:

- The arrival of a PDU through the dialogue service,
- The invocation of one of the service primitives by the user,
- The expiration of one of the internal timer, and
- The occurrence of an unrecoverable error.

4.4.4.3.3 The protocol is described under two forms: the textual and the tabular descriptions. The textual description takes precedence over the tabular description.

4.4.4.3.4 Unlike the ADS and FIS ASE, and like the CM ASE the CPDLC ASE is a singular entity.

4.4.4.3.5 There is a requirement that the ASE does not accept the invocation of primitives when no actions are described for that primitive in that state (CPDLC SARPs sections 2.3.5.3 and 2.3.5.5). Some explanation of this statement is needed. The CPDLC-ASE has several different states. When an ASE is in a particular state, only some primitives are permitted to be invoked. For example, if a CPDLC-start request is invoked, the CPDLC-user is not permitted to invoke a second CPDLC-start request on that connection (a confirmation has been received for the first one). There is no statement in the description of the protocol that explains what the ASE should do if it receives a second CPDLC-start request before the reply to the first one has been received. The SARPs therefore require that the CPDLC-user must not invoke a CPDLC-start request during this period.

4.4.4.3.6 Thus only actions which are permitted are described. If an action is not described, then it is not permitted.

4.4.4.3.7 Actions are enumerated in the order they have to be carried out by the protocol machine.

4.4.4.4 *CPDLC SARPs Section 2.3.5.7 CPDLC-ASE State Tables*

4.4.4.4.1 State tables are provided in SARPs section 2.3.5.7. These should be an exact reflection of the CPDLC protocol description, in a condensed form. The state tables are only presented for guidance, since the textual protocol description always takes precedence.

4.4.5 **CPDLC SARPs Section 2.3.6: Communication Requirements**

4.4.5.1 ***Encoding Rules***

4.4.5.1.1 This section states that PER (Packed Encoding Rules) must be used to encode the PDUs. PER is an ISO standard and is particularly efficient at encoding data. Implementors may use ASN.1 compilers to generate code that creates PER automatically.

4.4.5.2 ***Dialogue Service Requirements***

4.4.5.2.1 The dialogue service requires a number of parameters to operate. This section defines those parameters that are not defined elsewhere in this chapter, and states that the dialogue service must exhibit consistent behavior with the Upper Layer Communications [2].

4.4.5.2.2 The dialogue service requires *Quality of Service* (QOS) parameters. These parameters are: *Class of Communication*, *Application Priority*, and *Residual Error Rate.* The *Class of Communication* parameter is optionally provided by the CPDLC-user. If this parameter is not supplied by the CPDLC-user, it is provided by the dialogue service. The QOS values for *Application Priority* and the *Residual Error Rate* are fixed for CPDLC. Due to the fixed nature of these parameters they are supplied by the CPDLC-ASE and do not need to be supplied by the CPDLC-user.

4.4.5.2.2.1 The *Application Priority* is set by the CPDLC-ASE to the abstract value “high priority flight safety messages”.

4.4.5.2.2.2 The *Residual Error Rate* is set by the CPDLC-ASE to the abstract value “low”.

4.4.5.2.3 The dialogue service also has optional Security parameters: These parameters are not used by the CPDLC service.

4.4.5.2.4 The underlying layers provide a cyclic redundancy check and realize the requirements for message integrity. It is not necessary, therefore, to provide application level integrity by means such as redundancy checks and confirmation of services.

4.4.6 **CPDLC SARPs Section 2.3.7: CPDLC User Requirements**

4.4.6.1 ***Purpose of SARPs Section 2.3.7***

4.4.6.1.1 The requirements set out in this section are in line with those specified by [3]. These requirements are necessary for proper interoperability of ATN applications, and go into more technical detail than those specified by [3]. There are also several CPDLC-user requirements that are included in the CPDLC SARPs section 2.3.7 that go beyond strict interoperability, but ensure operational consistency between differing airspaces. These requirements are also in line with [3].

4.4.6.2 *Contents of SARPs Section 2.3.7*

4.4.6.2.1 The CPDLC SARPs section 2.3.7 cover four basic areas:

- CPDLC-user requirements in relationship to the CPDLC services outlined in SARPs section 2.3.3,
- CPDLC message handling requirements,
- CPDLC message element restrictions, and
- CPDLC message element tables, complementing the ASN.1, but also including the operational intent.

4.4.6.2.2 For CPDLC, in particular, it has been very difficult to determine just where to draw the line for inclusion of requirements on the CPDLC users. Additional requirements based on operational and procedural constraints may be placed on CPDLC users in addition to those in the CPDLC SARPs section 2.3.7, but are outside of the scope of this chapter.

4.4.6.3 *Response Time Requirements*

4.4.6.3.1 Other than the CPDLC-start and DSC-start response time requirements, there are no operational timer recommendations for when a response must be issued upon receipt of a CPDLC indication. The timer requirements (and a prohibition of message inclusion upon acceptance of a CPDLC or DSC start request) for a CPDLC and DSC start response are due to the technical constraint of not allowing the dialogue requester to invoke any other (except aborts) CPDLC or DSC primitives until a confirmation is received for a specific dialogue. Operationally this time must be minimized as much a possible.

4.4.6.3.2 No other operational response times are established by the SARPs. All other operational response time values will be determined by operational and procedural requirements. No implicit aborts (i.e., a SARPs required CPDLC-user-abort, a CPDLC-provider-abort, a D-U-ABORT, or a D-ABORT) will occur due to the failure of any operational timer. (A local parameter could be set up to allow a given system to issue a CPDLC-user-abort due to operational timer failure.)

4.4.7 CPDLC SARPs Section 2.3.8: Subsetting Rules

4.4.7.1 There is some functionality within the CPDLC SARPs that implementors may choose not to incorporate. For example, a particular implementation of an CPDLC-airborne application does not have to implement the DSC capability.

4.4.7.2 There are some combinations of functionality that allow interoperability, and some do not. The CPDLC SARPs section 2.3.8 defines the combinations of functionality that allow interoperability.

4.4.7.3 The options are defined in a set of tables.

- Version number — only one version is defined. This is a place holder for when future versions are defined.

- Protocol Options — this defines a number of options for parts of the protocol that may be implemented. The options may be implemented together. Each has a name associated with it — the predicate.

- CPDLC-ground-ASE configurations — this defines eight combinations of protocol options, each of which yields a coherent protocol.

- CPDLC-air-ASE configurations — this defines two combinations of protocol options, each of which yields a coherent protocol.

- Supported CPDLC service primitives — this defines the conditions under which the service primitives are applicable.

- Supported CPDLC APDUs — this defines the conditions under which the PDUs are applicable.

4.4.7.4 Functionality supported by an air-based CPDLC system is different from the functionality supported by a ground-based CPDLC system.

4.4.7.5 Any aircraft can independently select a subset and be inter-operable with any ground system independently selecting a subset.

4.4.7.6 *Mandatory Functionality*

4.4.7.6.1 All CPDLC systems must support the core CPDLC functionality which includes the CPDLC-start, CPDLC-message, CPDLC-end plus abort services.

4.4.7.6.2 A CPDLC-air system is not required to support any other (i.e. DSC) functionality.

4.4.7.6.3 A CPDLC-ground system must also at a minimum support:

- the capability to receive and reject a request for a DSC dialogue

- the capability to receive a request for the forward service and indicate that the forward service is not supported.

4.4.7.7 *Optional Functionality*

4.4.7.7.1 The CPDLC-air system may optionally support the DSC service.

4.4.7.7.2 The CPDLC-ground system may optionally support:

- full DSC capability

- CPDLC-ground forward initiator capability,

- full CPDLC-ground forward receiver capability.

4.4.7.8 The various optional capabilities can be implemented totally independent of one another.

4.5 Dimensions

4.5.1 PDU Size

4.5.1.1 Table 4.5-1 provides the sizes of typical CPDLC messages.

Table 4.5-1. Typical CPDLC Message Sizes

Exchange Type	Message Type	Typical Size (Octets)	Comments
Downlink (AircraftPDUs)	abortUser	1	
	abortProvider	1	
	startdown	2	Mode, NULL message (no header)
	startdown	14	Mode, Message Header, Downlink Message 20
	send	13	Message header, Downlink Message 0 (WILCO)
Uplink (GroundPDUs)	abortUser	1	
	abortProvider	1	
	startup	1	NULL message (no header)
	startup	15	Message header, Uplink Message 20, level = 350
	send	45	Message header, Uplink Message 20, Uplink Message 165, Uplink Message 52 [position] is Navaid + lat/lon, Uplink Message 78 [position] is in Lat/Lon DegreesMinutes, Uplink Message 93
	send	1607	Message header, Uplink Message 85 (RouteClearance has 20 RouteInformation elements, RouteInformation Additional field included also)

Exchange Type	**Message Type**	**Typical Size (Octets)**	**Comments**
Ground-ground (GroundPDUs)	forward	19	Forward Message header, Downlink Message 18
	forwardresponse	1	

4.5.2 **Rate of Message Frequency**

4.5.2.1 The rate of CPDLC message exchange is dependent on several factors which include, but are not limited to, the following:

- the operational domain (i.e., terminal, en-route, oceanic),
- the instantaneous aircraft count,
- the percent of pilot-controller messages exchanged using CPDLC data vs. voice
- availability of ground-ground connectivity
- types of CPDLC service implemented (e.g., departure clearance, transfer of control, frequency assignments)
- availability of ADS (i.e., position data, extended projected profiles),
- level of automation, and
- Human Machine Interface (HMI) capabilities/constraints.

4.5.3 **Number of CPDLC/DSC Dialogues/Connections**

4.5.3.1 ***Airborne CPDLC/DSC Dialogue Requirements***

4.5.3.1.1 **Total CPDLC/DSC Connections**

4.5.3.1.1.1 The airborne CPDLC application must be capable of instantaneously supporting a total of up to 5 CPDLC dialogues if DSC is not supported, or up to 6 CPDLC or DSC dialogues, if DSC is supported, (ASE instanteations) as follows:

a) 2 simultaneously with the CDA:

1) in the situation that results from both an aircraft and a ground system in initiating a CPDLC dialogue simultaneously; the aircraft then subsequently terminates (aborts) one of the established dialogues, and the remaining dialogue is used for operational communication, and

2) in the situation that an aircraft receives a second request from its CDA to establish a CPDLC dialogue; this request is accepted and the aircraft then

terminates the previously established dialogue. This is to allow operational continuation of a CDA dialogue although a technical problem may have occurred with the first connection.

b) 2 simultaneously with the NDA

1) in the situation that results from both an aircraft and a ground system in initiating a CPDLC dialogue simultaneously; the aircraft then subsequently terminates (aborts) one of the established dialogues, and the remaining dialogue remains open but is not used operationally until when and if the NDA becomes the CDA, and

2) in the situation that an aircraft receives a second request from its NDA to establish a CPDLC dialogue, this request is accepted and the aircraft then terminates the previously established dialogue. This is to allow operational continuation of a NDA dialogue although a technical problem may have occurred with the first connection.

c) 1 ground request

1) the aircraft has to be able to receive a CPDLC dialogue start request from any other ground system that is not the CDA or NDA and then reject such a request.

d) 1 with a DDA (if supported by aircraft)

1) only the aircraft can initiate a DSC dialogue, thus the above situations requiring 2 simultaneous connection with the CDA and NDA do not occur with the DSC. The capability to do DSC is an airborne option.

4.5.3.1.2 Connections With a Given Ground System Peer

4.5.3.1.2.1 The airborne CPDLC application must be capable of simultaneously supporting a maximum of 3 CPDLC or DSC dialogues with a given ground system peer as follows:

a) 2 simultaneously with the CDA or 2 simultaneously with theNDA (a given ground system is never simultaneously a CDA and NDA) as described above, and

b) 1 with a DDA (aircraft can only have 1 DSC at a time)

4.5.3.2 *Ground CPDLC/DSC Dialogue Requirements*

4.5.3.2.1 Total Connections

4.5.3.2.1.1 The total number of CPDLC/DSC connections (ASE instanteations) a given ground system must be able to simultaneously support cannot be determined here. A given ground system's connections are determined by how many aircraft and other CPDLC ground systems it is in communication with simultaneously. This will be a mix of how many aircraft it is connected to as a CDA, how many other aircraft it is connected to as a NDA, how many aircraft it is connected to as a DSC, and how many other CPDLC ground system it is connected to.

4.5.3.2.2 **Connections With a Given Aircraft System Peer**

4.5.3.2.2.1 (Same as above under Airborne = 3)

4.5.3.2.3 **Connections With a Given Other Ground System Peer**

A ground system must be capable of supporting a maximum of two CPDLC connections with any other ground system peer as follows: 1 connection with a ground system with which it initiates a CPDLC dialogue, and one connection with that same ground system when it receives a request to initiate a CPDLC dialogue.

4.6 ASN.1 Index

4.6.1 ASN.1 Hierarchy

4.6.1.1 The following figures illustrate the hierarchy (both up and down) for each of the CPDLC ASN.1 variables.

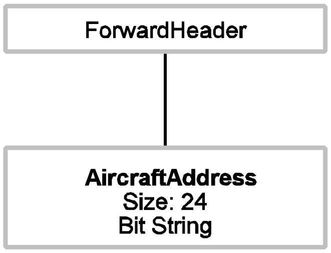


Figure 4.6-1. AircraftAddress Variable Hierarchy

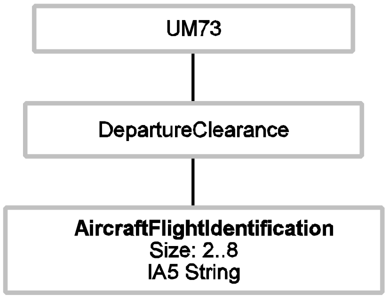


Figure 4.6-2. AircraftFlightIdentification Variable Hierarchy

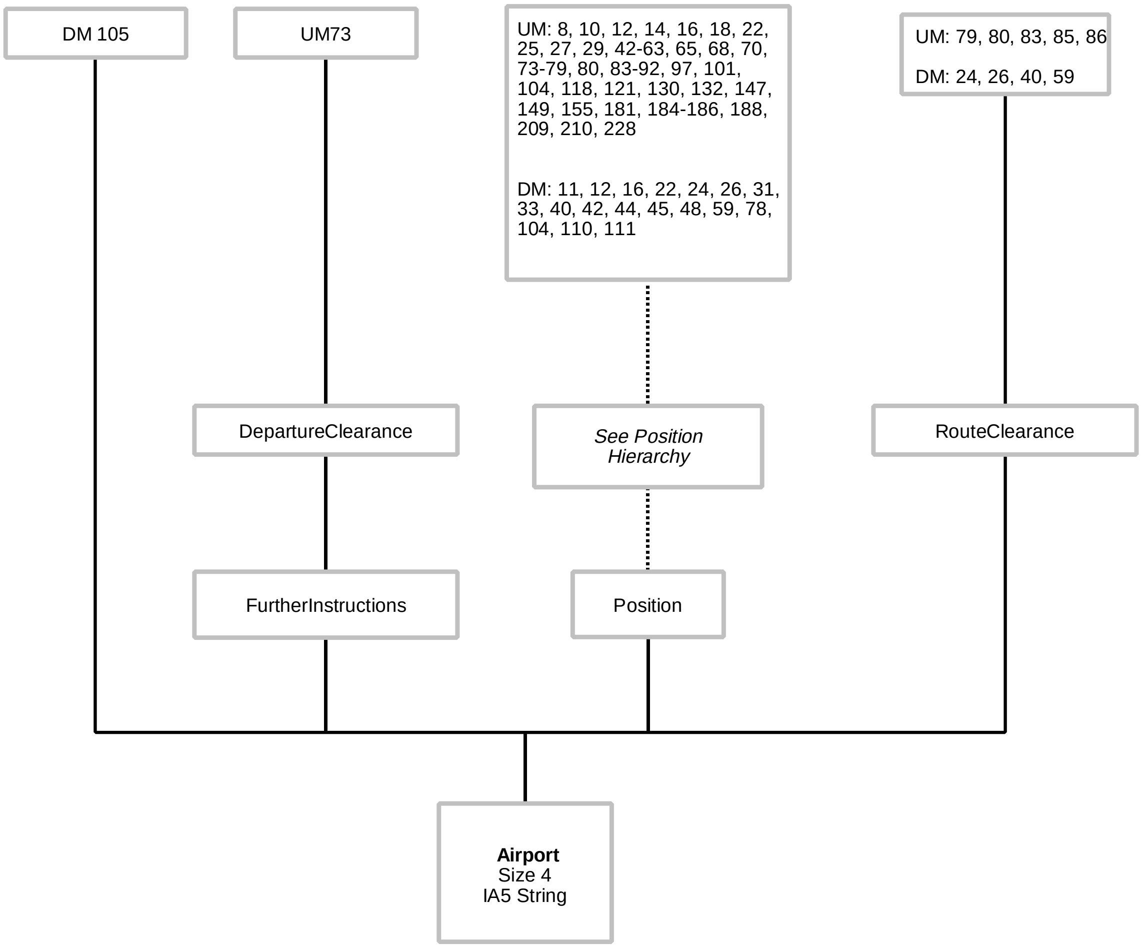

DM 105
UM73
UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 73-79, 80, 83-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 24, 26, 31, 33, 40, 42, 44, 45, 48, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DepartureClearance
See Position Hierarchy
RouteClearance
FurtherInstructions
Position
Airport
Size 4
IA5 String

Figure 4.6-3. Airport Variable Hierarchy

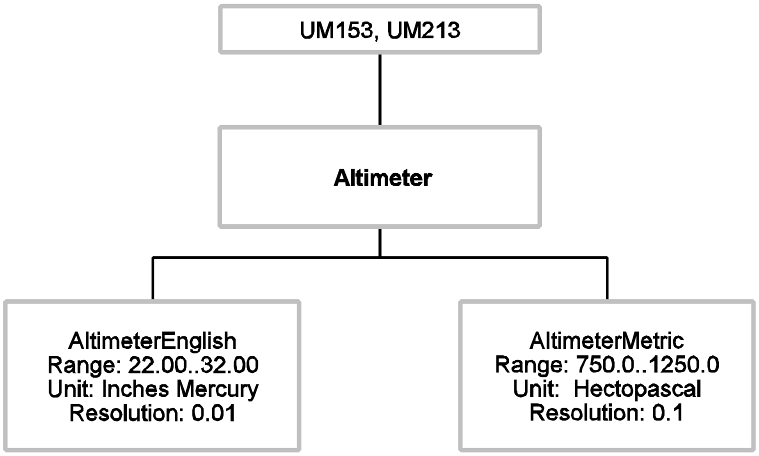


Figure 4.6-4. Altimeter Variable Hierarchy

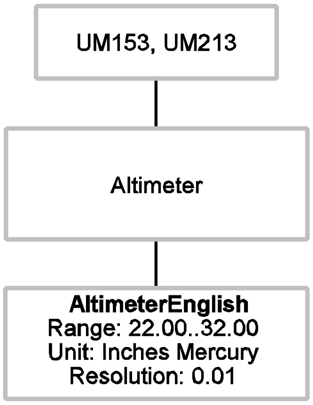


Figure 4.6-5. AltimeterEnglish Variable Hierarchy

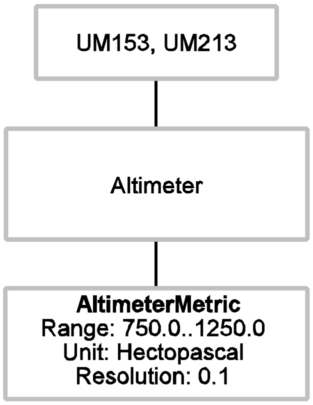


Figure 4.6-6. AltimeterMetric Variable Hierarchy

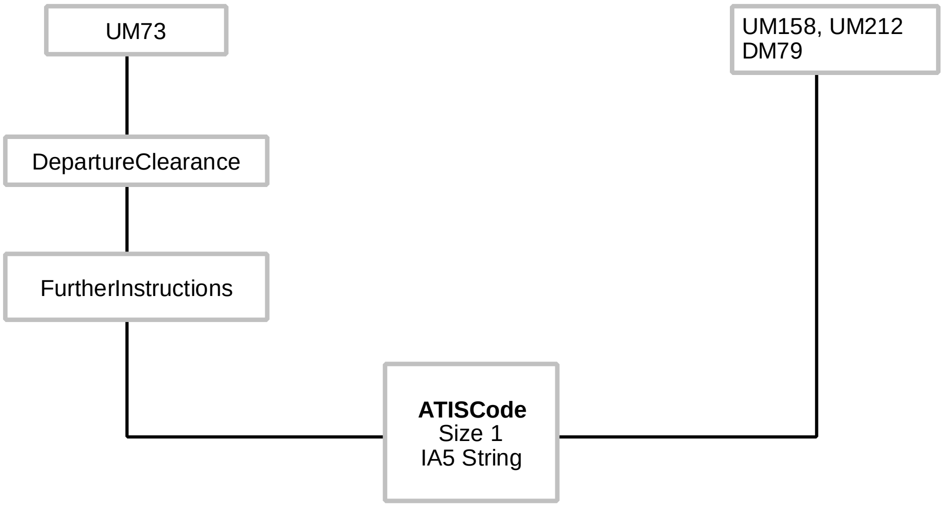


Figure 4.6-7. ATISCode Variable Hierarchy

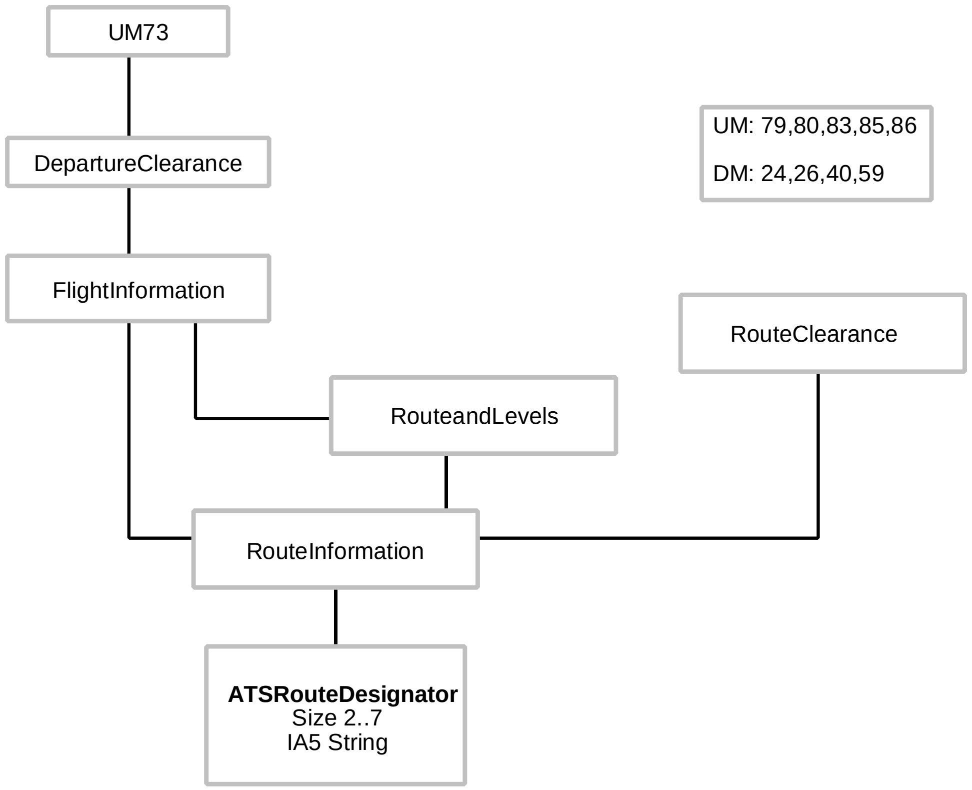


Figure 4.6-8. ATSRouteDesignator Variable Hierarchy

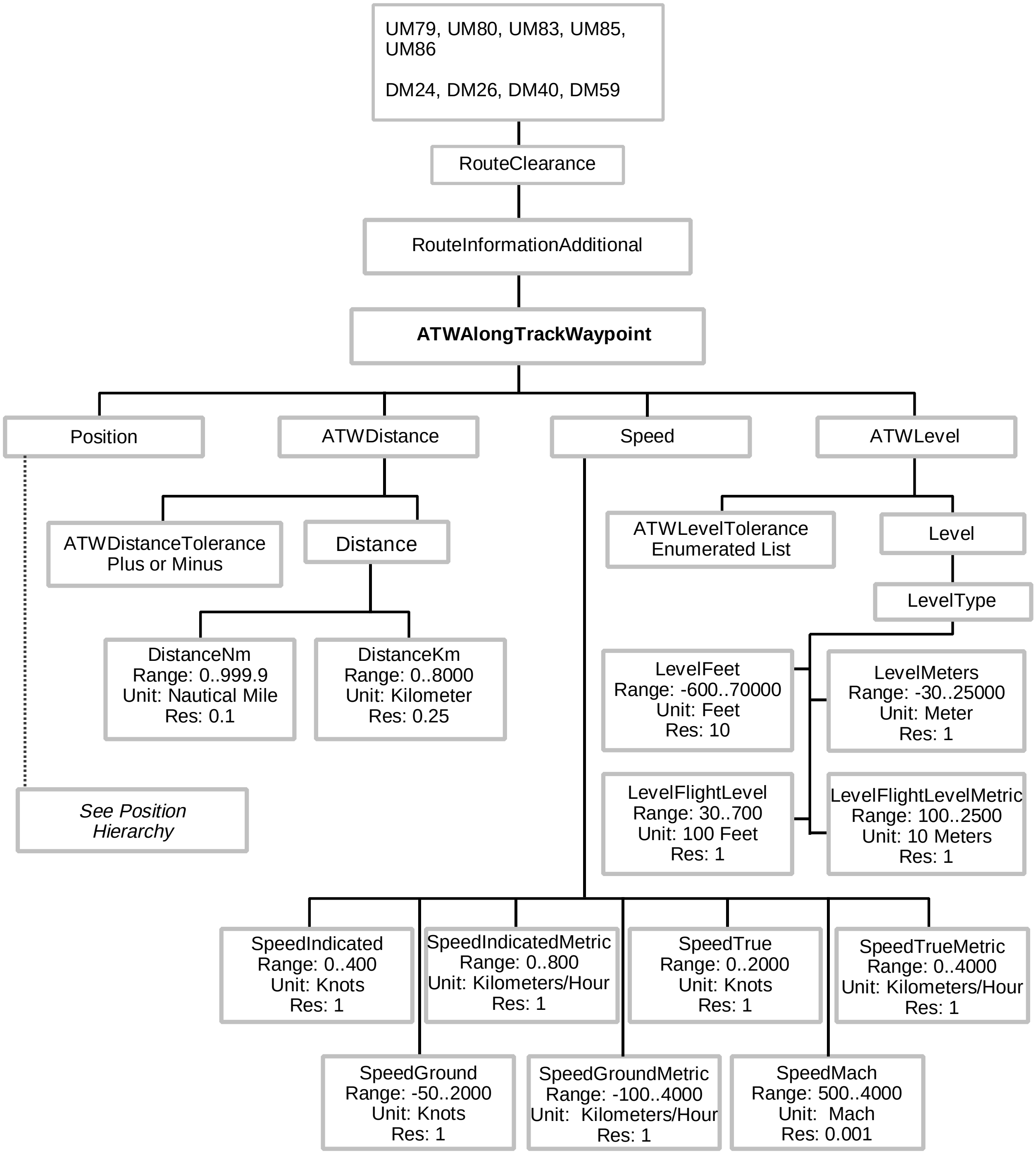

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
Position
ATWDistance
Speed
ATWLevel
ATWDistanceTolerance Plus or Minus
Distance
DistanceNm Range: 0..999.9 Unit: Nautical Mile Res: 0.1
DistanceKm Range: 0..8000 Unit: Kilometer Res: 0.25
See Position Hierarchy
ATWLevelTolerance Enumerated List
Level
LevelType
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
SpeedIndicated Range: 0..400 Unit: Knots Res: 1
SpeedIndicatedMetric Range: 0..800 Unit: Kilometers/Hour Res: 1
SpeedTrue Range: 0..2000 Unit: Knots Res: 1
SpeedTrueMetric Range: 0..4000 Unit: Kilometers/Hour Res: 1
SpeedGround Range: -50..2000 Unit: Knots Res: 1
SpeedGroundMetric Range: -100..4000 Unit: Kilometers/Hour Res: 1
SpeedMach Range: 500..4000 Unit: Mach Res: 0.001

Figure 4.6-9. ATWAlongTrackWaypoint Variable Hierarchy

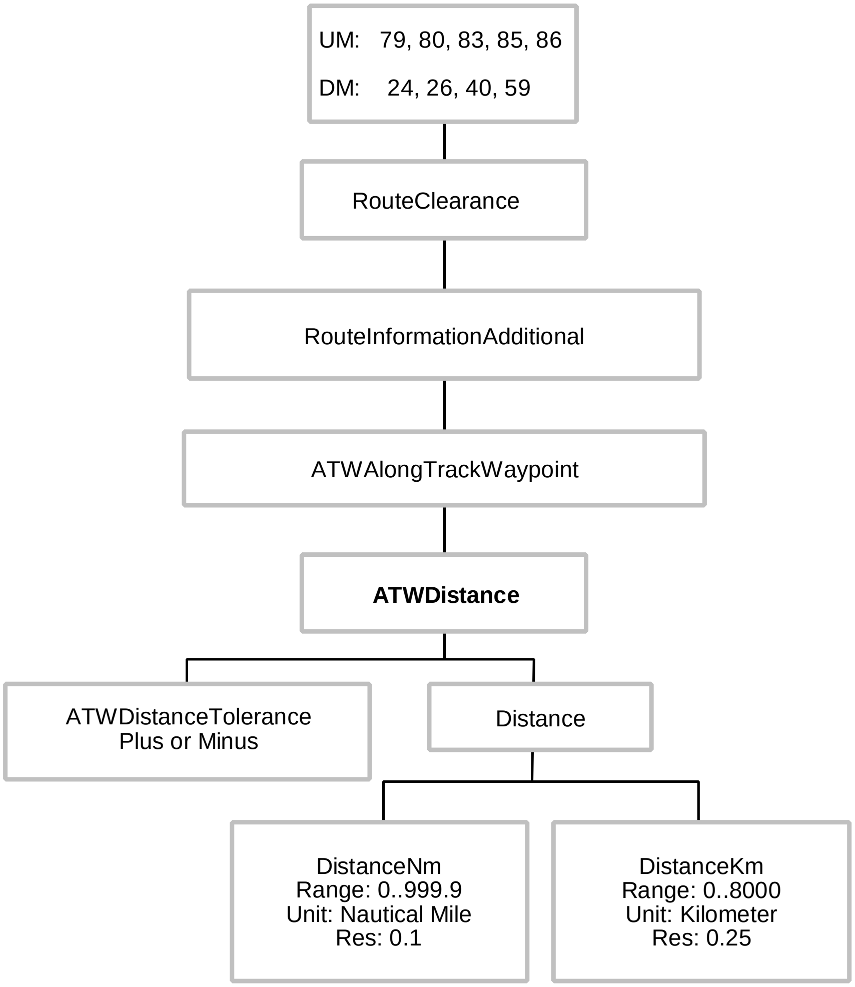


Figure 4.6-10. ATWDistance Variable Hierarchy

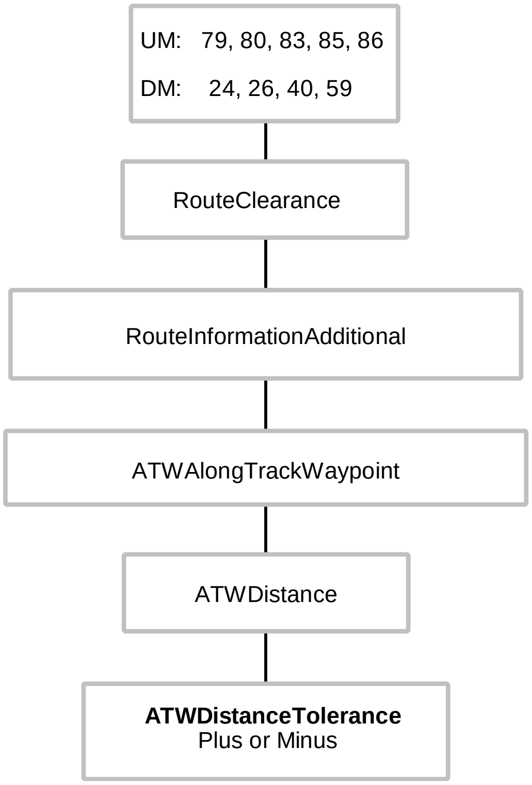


Figure 4.6-11. ATWDistanceTolerance Variable Hierarchy

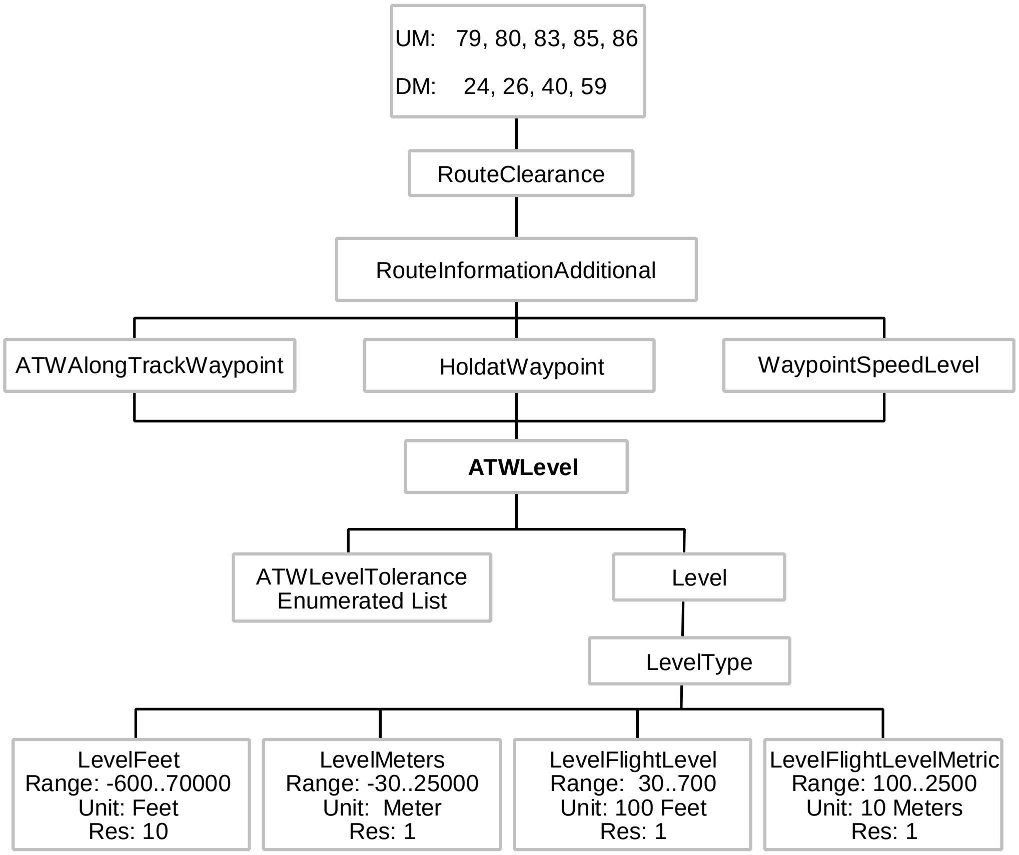

UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
RouteClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldatWaypoint
WaypointSpeedLevel
ATWLevel
ATWLevelTolerance Enumerated List
Level
LevelType
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1

Figure 4.6-12. ATWLevel Variable Hierarchy

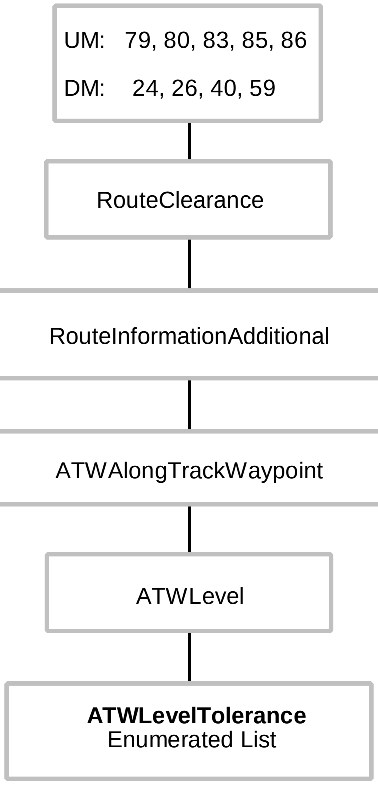


Figure 4.6-13. ATWLevelTolerance Variable Hierarchy

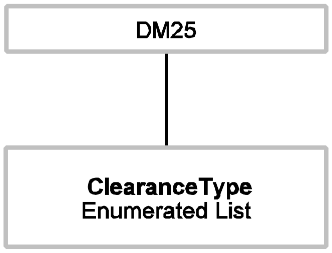


Figure 4.6-14. ClearanceType Variable Hierarchy

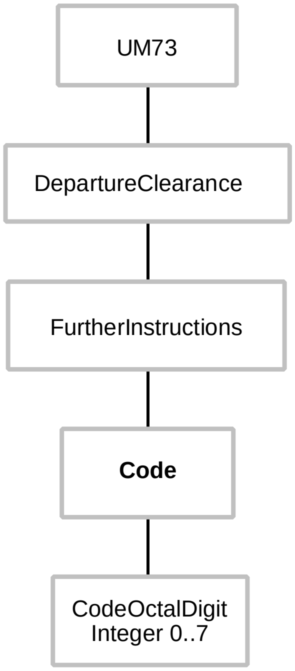


Figure 4.6-15. Code Variable Hierarchy

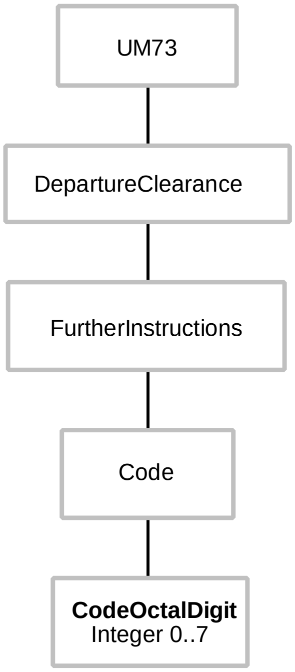


Figure 4.6-16. CodeOctalDigit Variable Hierarchy

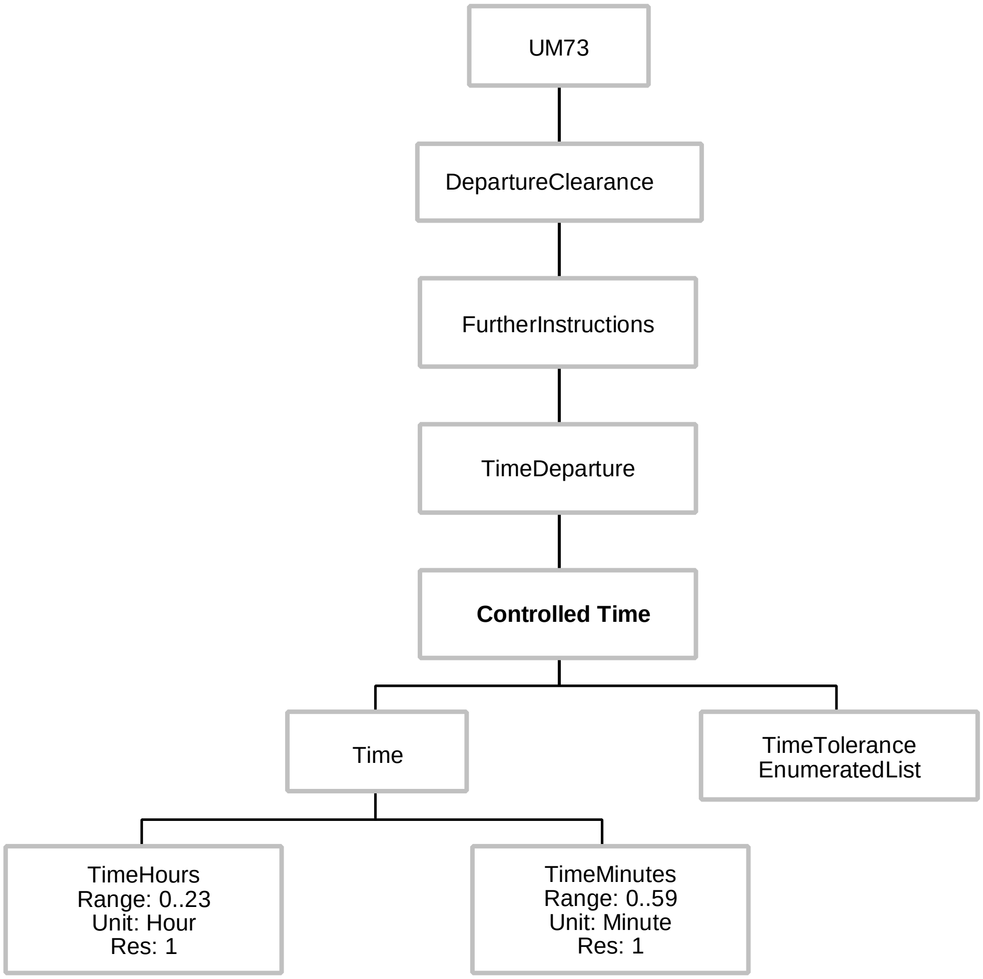


Figure 4.6-17. ControlledTime Variable Hierarchy

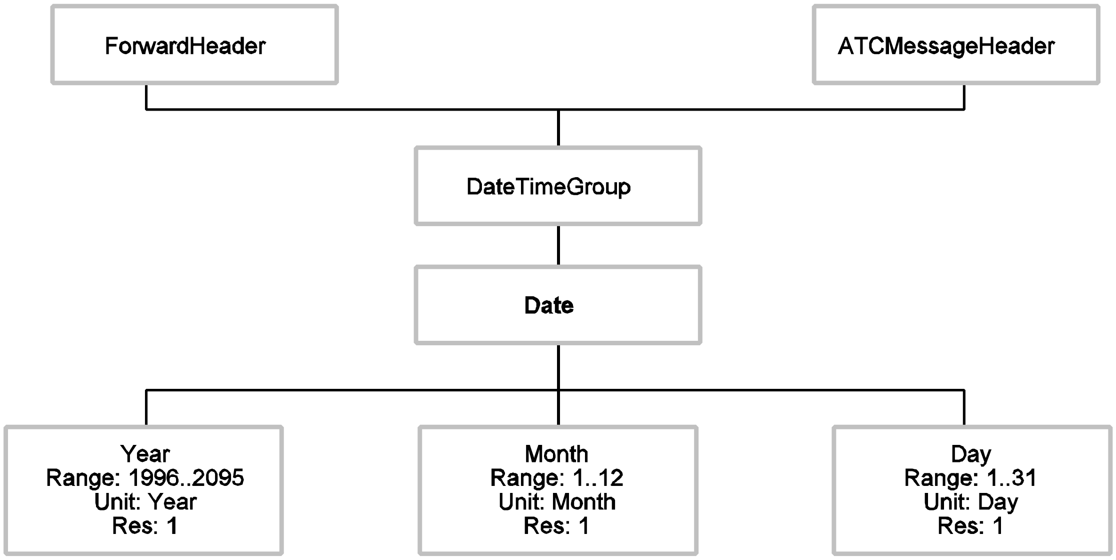


Figure 4.6-18. Date Variable Hierarchy

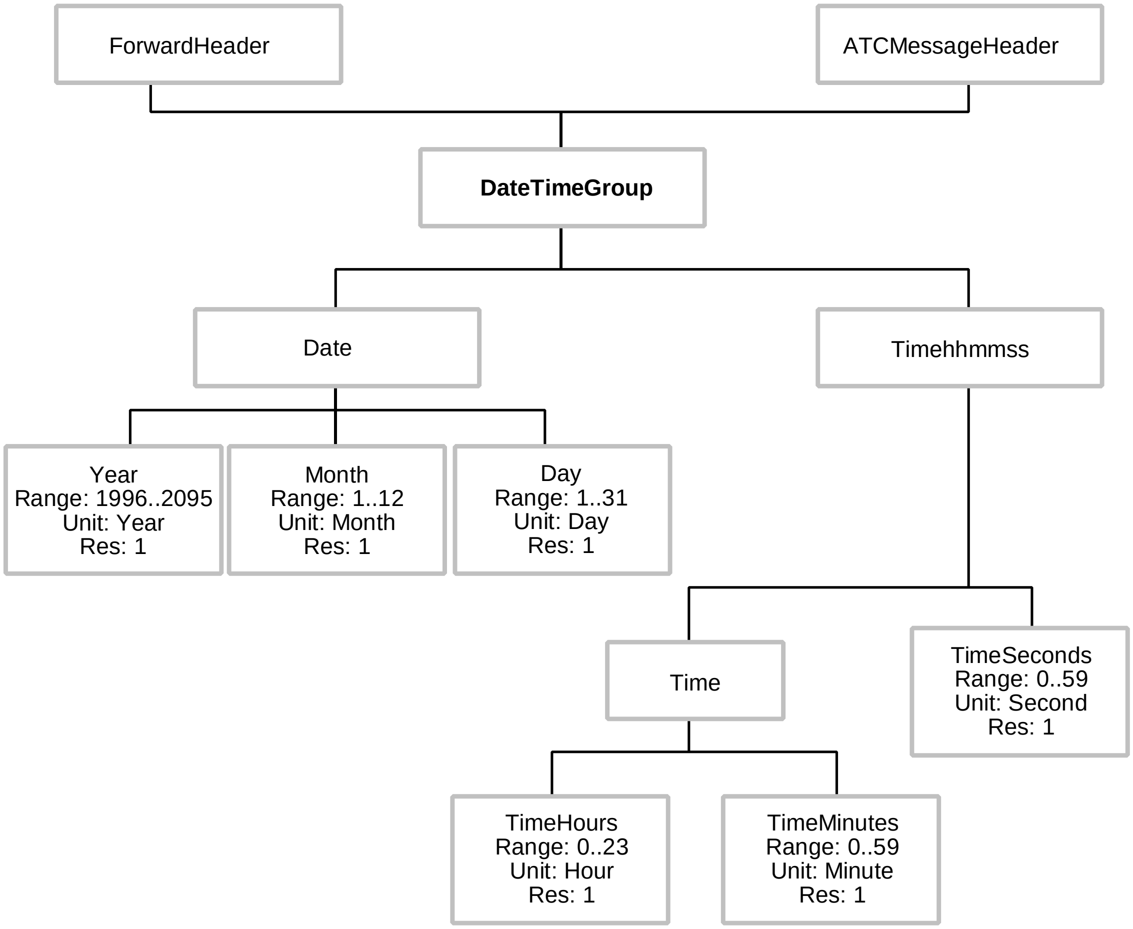


Figure 4.6-19. DateTimeGroup Variable Hierarchy

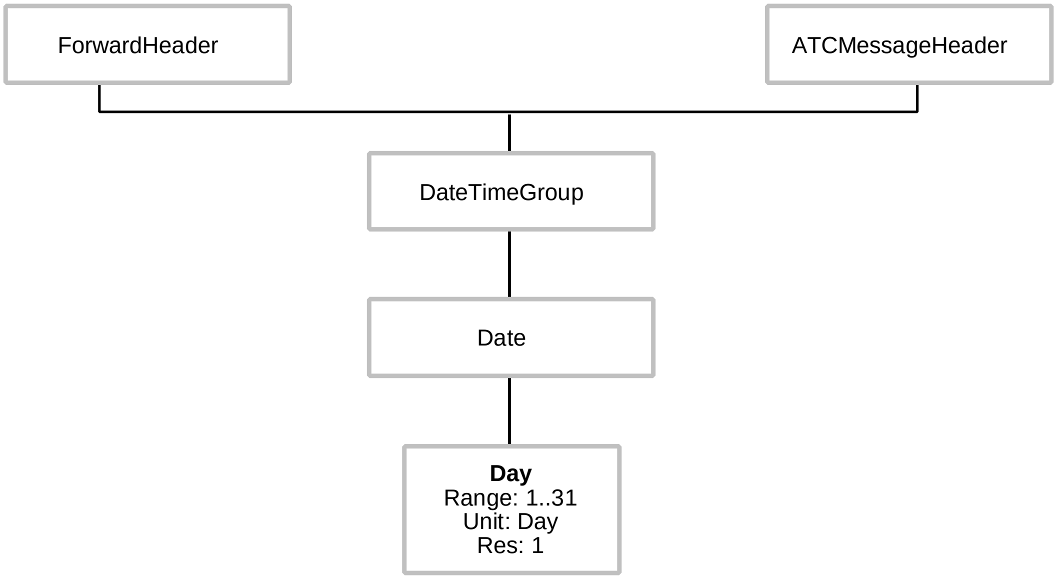


Figure 4.6-20. Day Variable Hierarchy

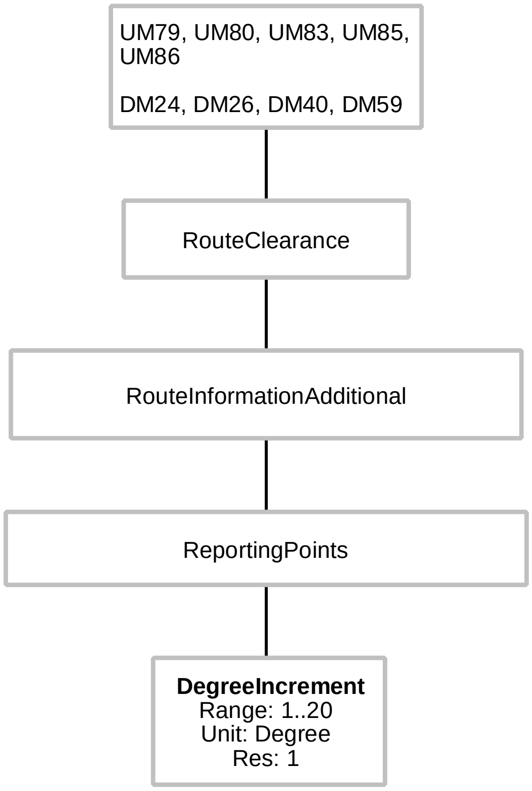


Figure 4.6-21. DegreeIncrement Variable Hierarchy

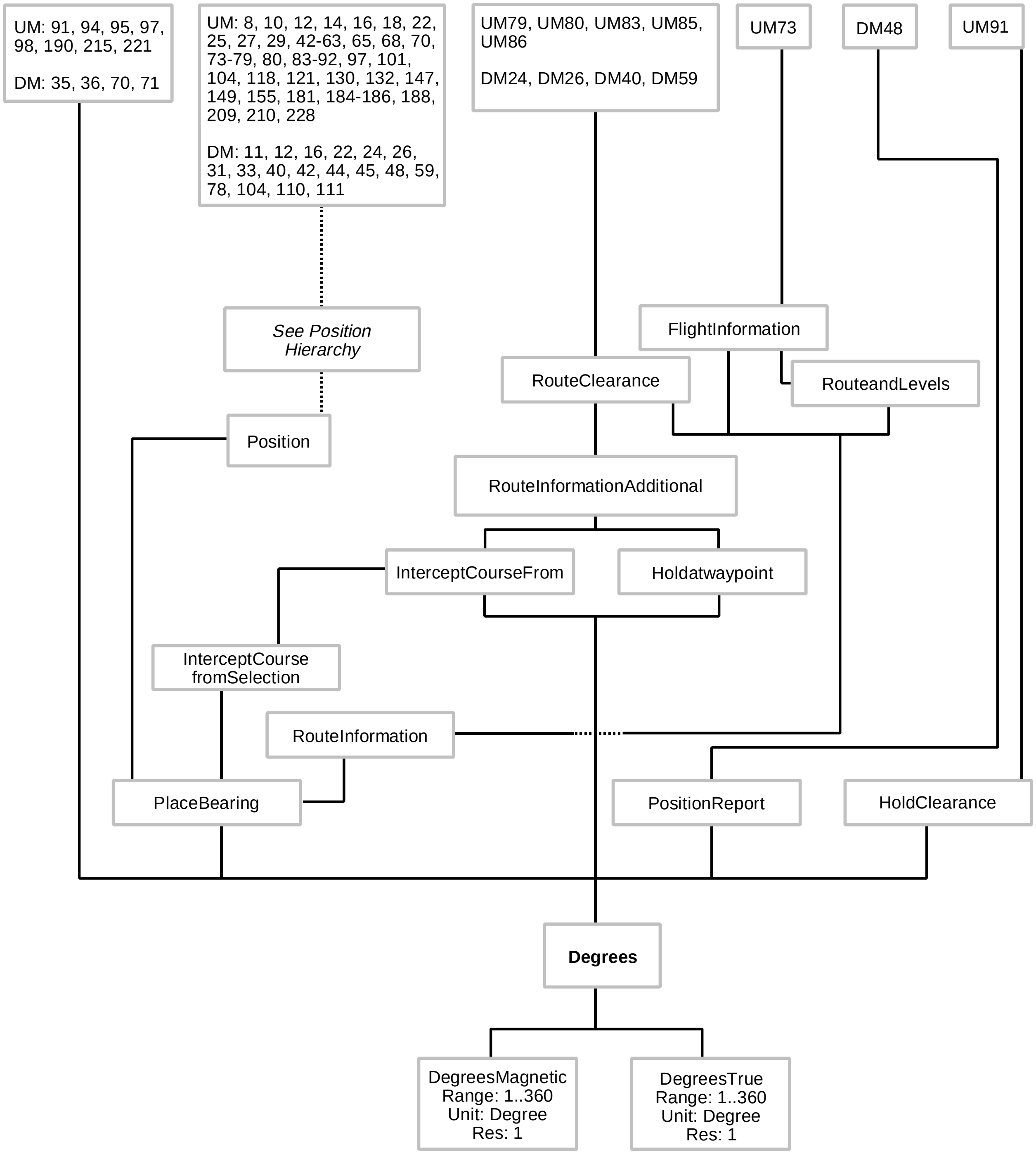

UM: 91, 94, 95, 97, 98, 190, 215, 221
DM: 35, 36, 70, 71
UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 73-79, 80, 83-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 24, 26, 31, 33, 40, 42, 44, 45, 48, 59, 78, 104, 110, 111
UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM73
DM48
UM91
See Position Hierarchy
FlightInformation
RouteClearance
RouteandLevels
Position
RouteInformationAdditional
InterceptCourseFrom
Holdatwaypoint
InterceptCourse fromSelection
RouteInformation
PlaceBearing
PositionReport
HoldClearance
Degrees
DegreesMagnetic
Range: 1..360
Unit: Degree
Res: 1
DegreesTrue
Range: 1..360
Unit: Degree
Res: 1

Figure 4.6-22. Degrees Variable Hierarchy

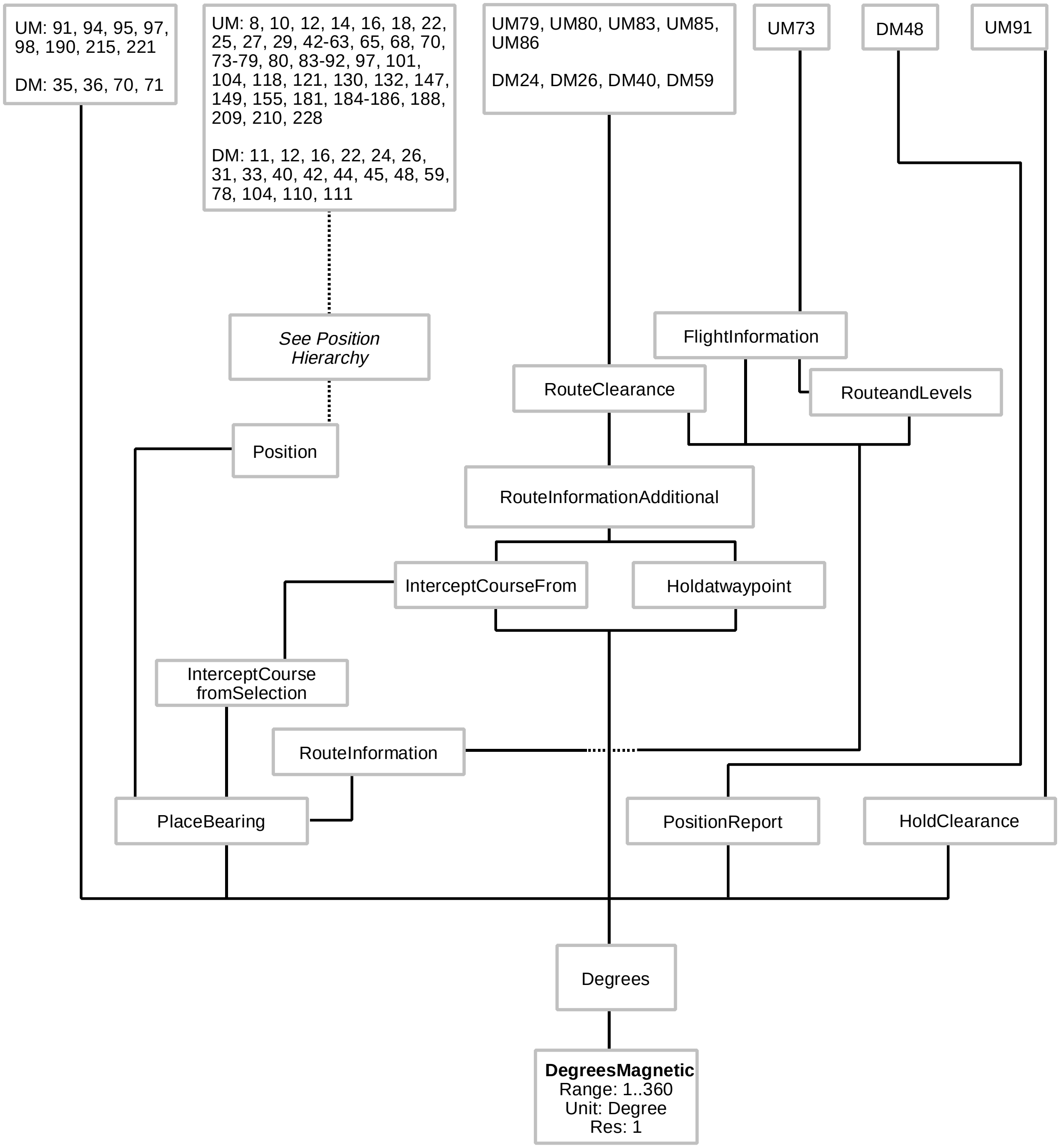

UM: 91, 94, 95, 97, 98, 190, 215, 221
DM: 35, 36, 70, 71
UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 73-79, 80, 83-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 24, 26, 31, 33, 40, 42, 44, 45, 48, 59, 78, 104, 110, 111
UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM73
DM48
UM91
See Position Hierarchy
FlightInformation
RouteClearance
RouteandLevels
Position
RouteInformationAdditional
InterceptCourseFrom
Holdatwaypoint
InterceptCourse fromSelection
RouteInformation
PlaceBearing
PositionReport
HoldClearance
Degrees
DegreesMagnetic
Range: 1..360
Unit: Degree
Res: 1

Figure 4.6-23. DegreesMagnetic Variable Hierarchy

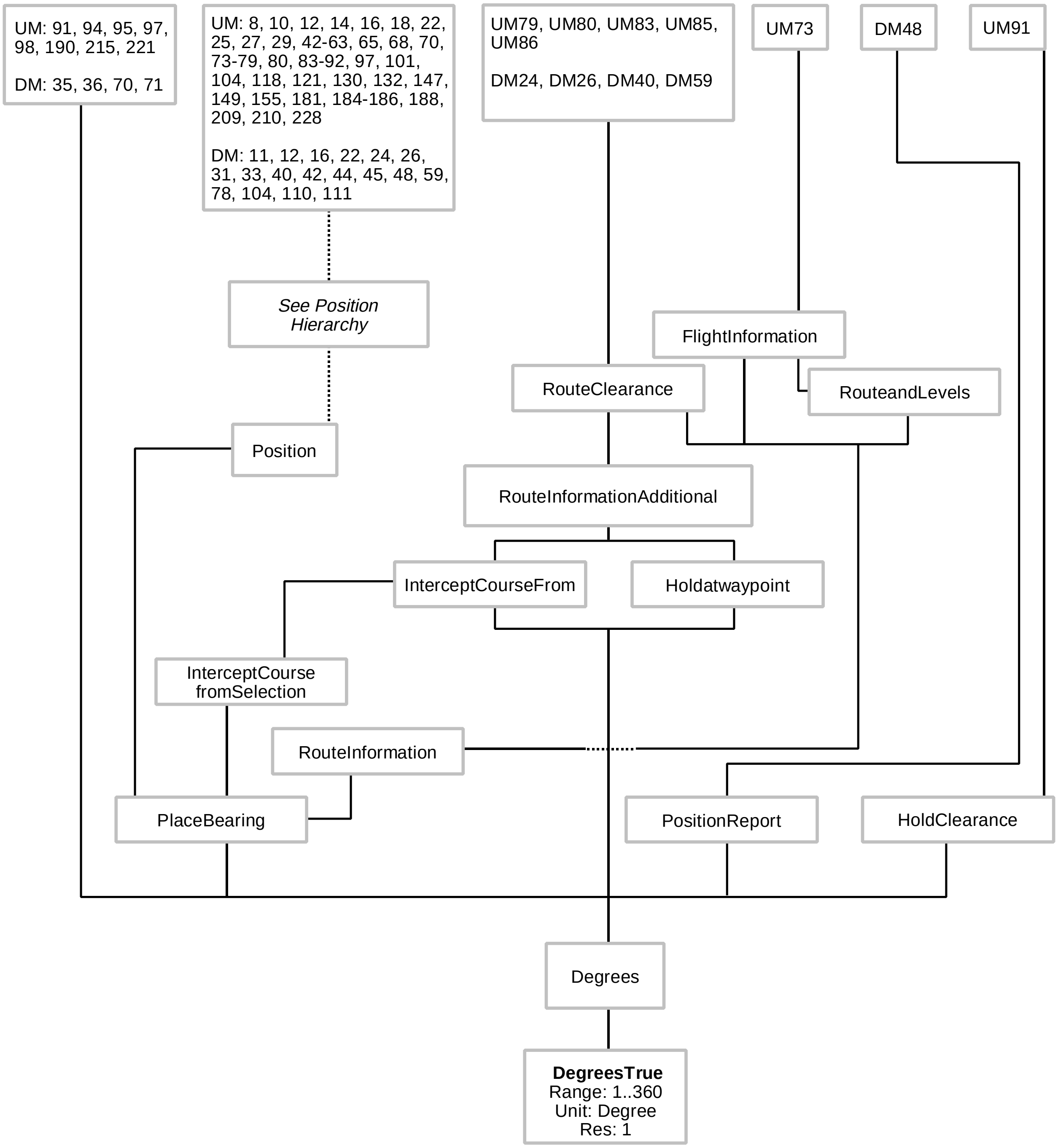

UM: 91, 94, 95, 97, 98, 190, 215, 221
DM: 35, 36, 70, 71
UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 73-79, 80, 83-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 24, 26, 31, 33, 40, 42, 44, 45, 48, 59, 78, 104, 110, 111
UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM73
DM48
UM91
See Position Hierarchy
FlightInformation
RouteClearance
RouteandLevels
Position
RouteInformationAdditional
InterceptCourseFrom
Holdatwaypoint
InterceptCourse fromSelection
RouteInformation
PlaceBearing
PositionReport
HoldClearance
Degrees
DegreesTrue
Range: 1..360
Unit: Degree
Res: 1

Figure 4.6-24. DegreesTrue Variable Hierarchy

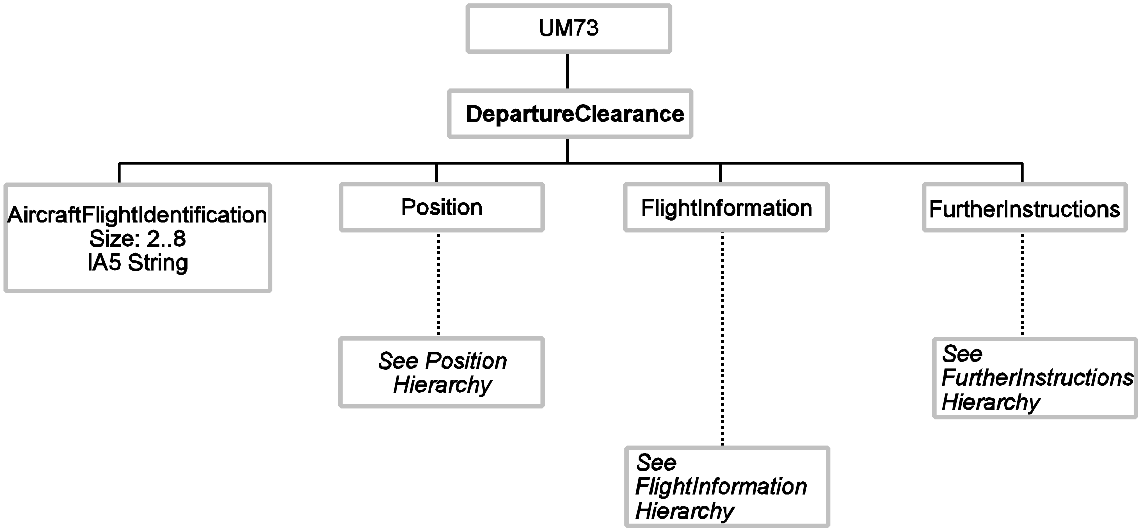


Figure 4.6-25. DepartureClearance Variable Hierarchy

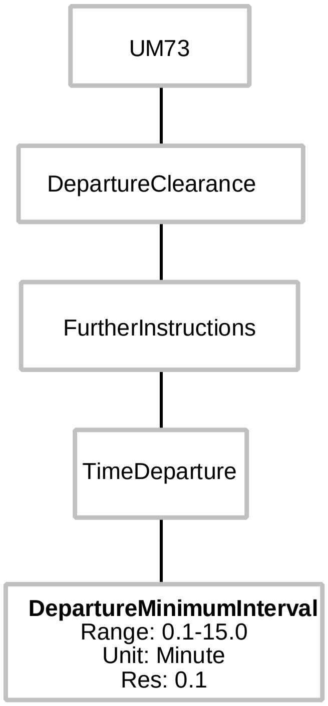


Figure 4.6-26. DepartureMinimumInterval Variable Hierarchy

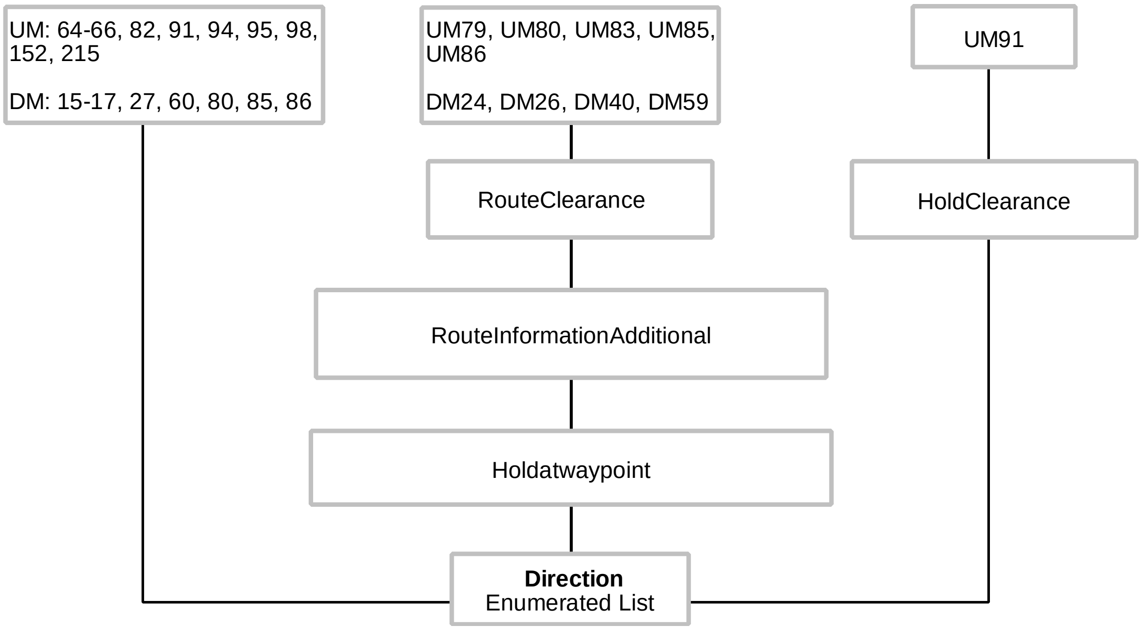

UM: 64-66, 82, 91, 94, 95, 98, 152, 215
DM: 15-17, 27, 60, 80, 85, 86
UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM91
RouteClearance
HoldClearance
RouteInformationAdditional
Holdatwaypoint
Direction
Enumerated List

Figure 4.6-27. Direction Variable Hierarchy

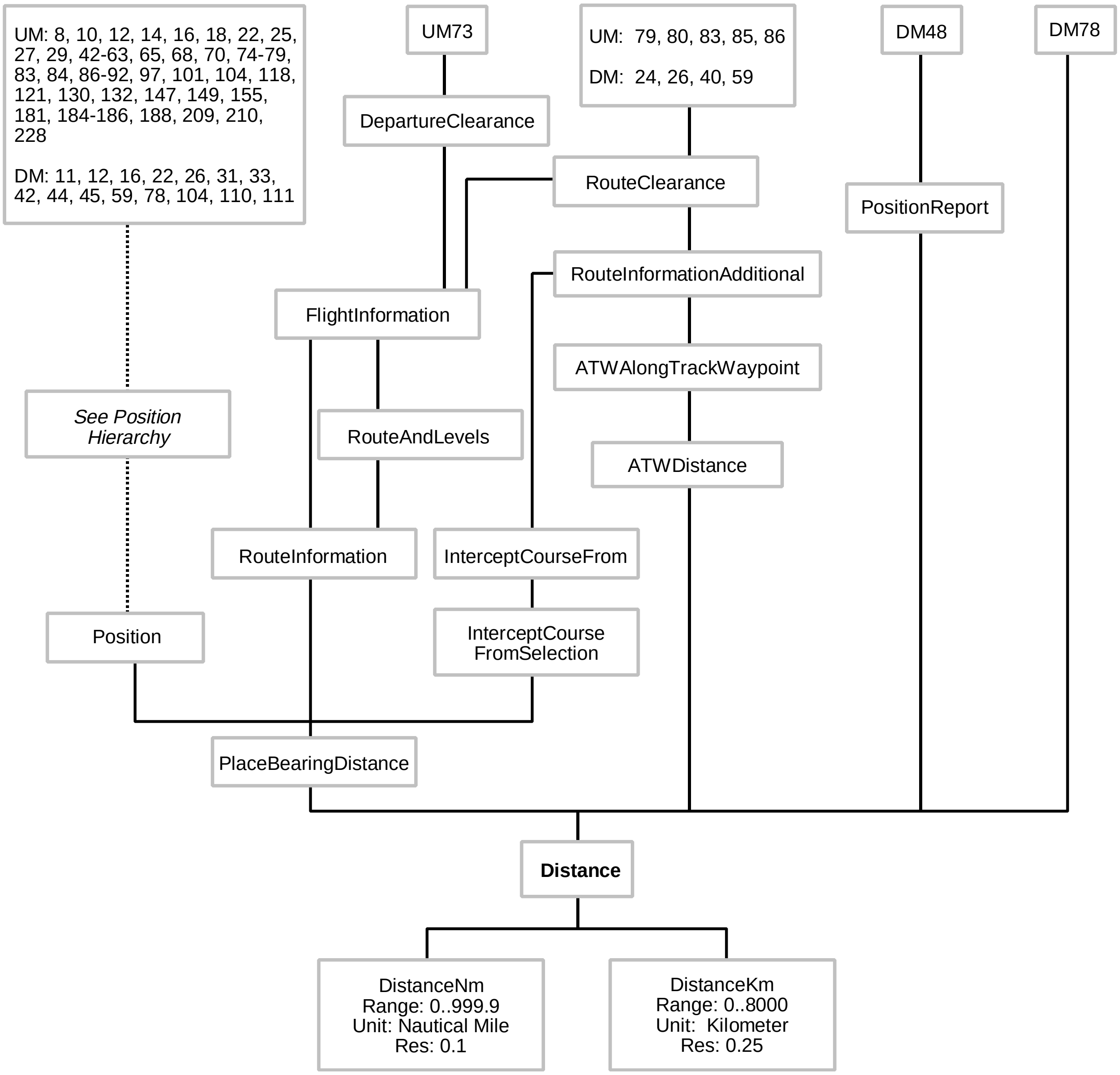


Figure 4.6-28. Distance Variable Hierarchy

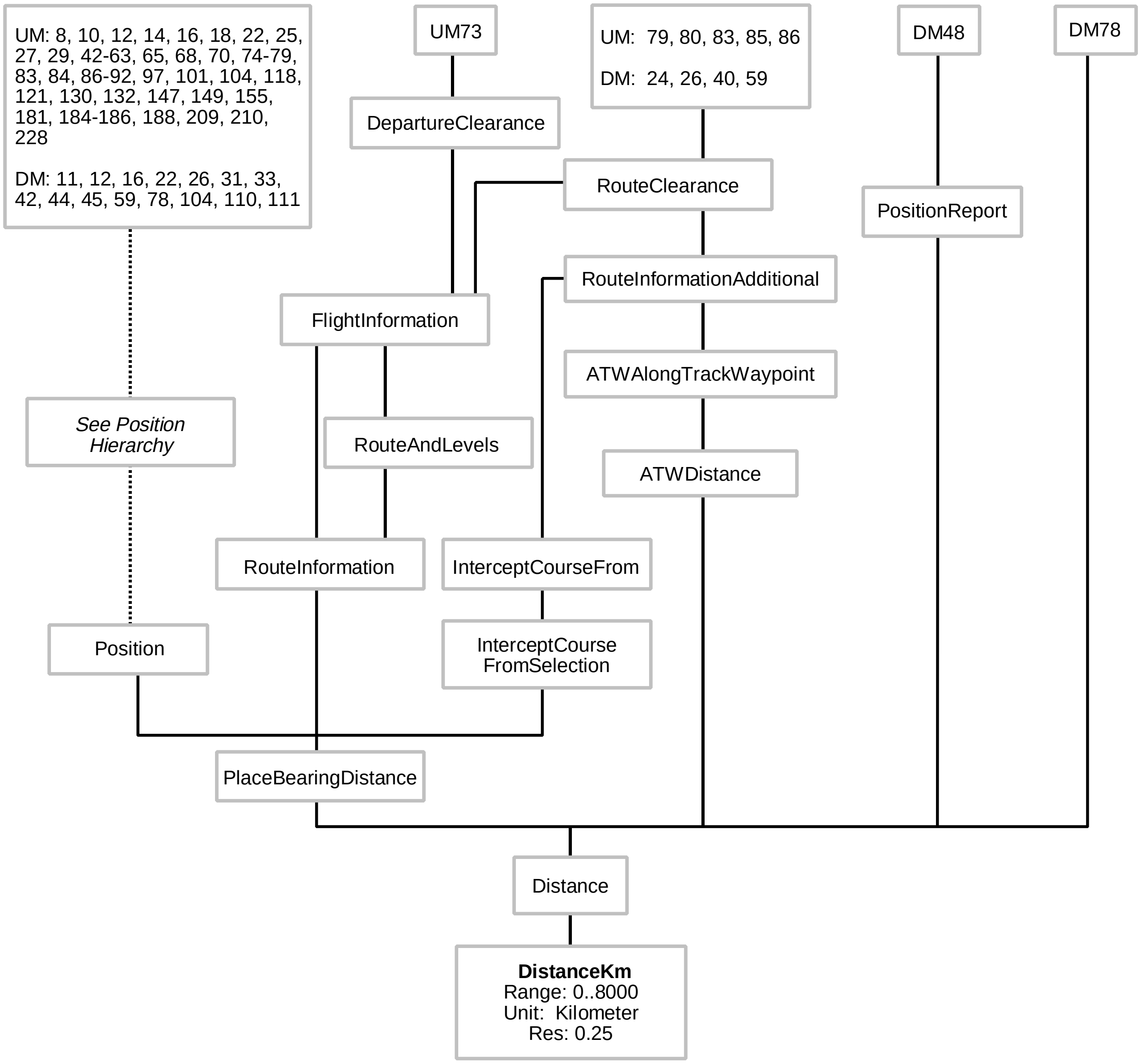


Figure 4.6-29. DistanceKm Variable Hierarchy

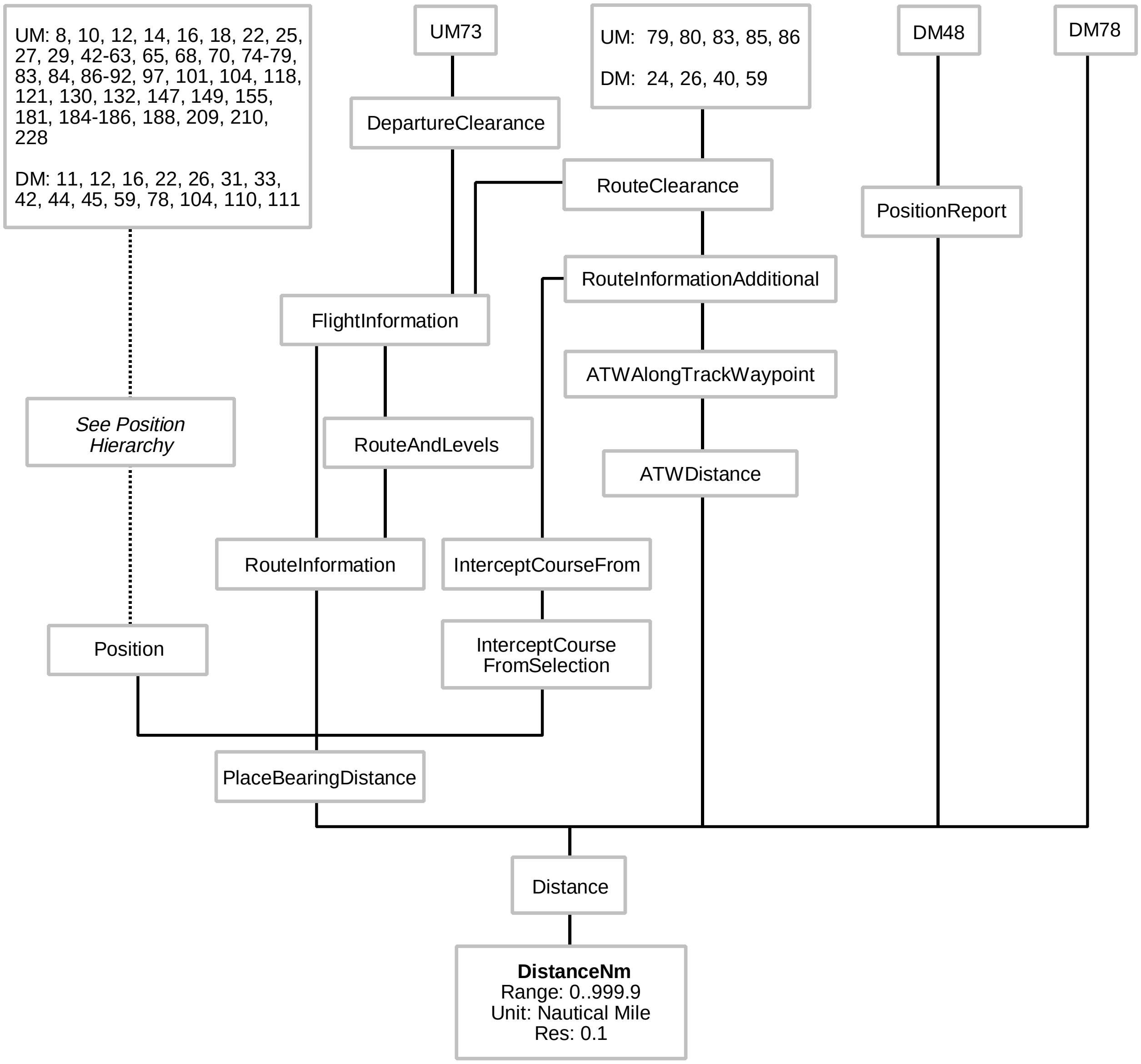


Figure 4.6-30. DistanceNm Variable Hierarchy

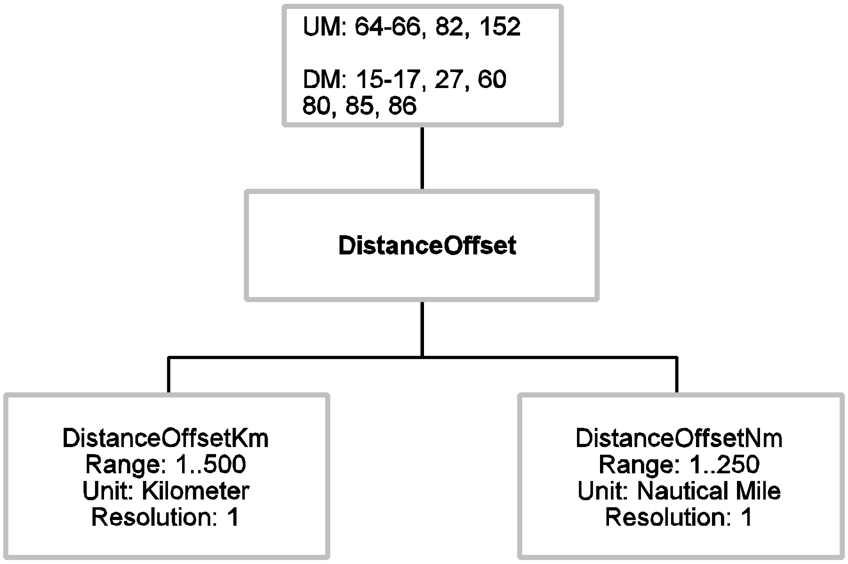


Figure 4.6-31. DistanceOffset Variable Hierarchy

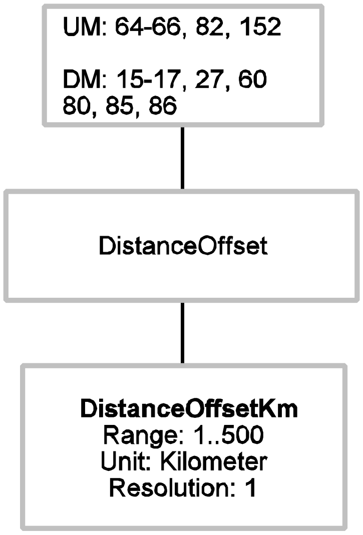


Figure 4.6-32. DistanceOffsetKm Variable Hierarchy

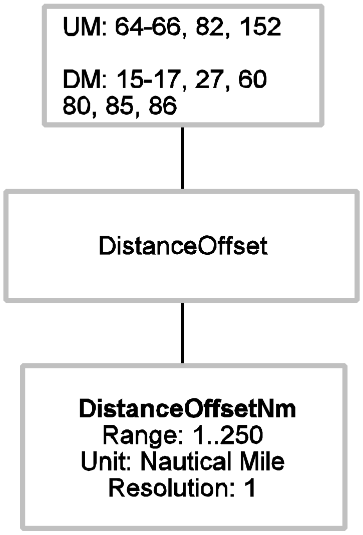


Figure 4.6-33. DistanceOffsetNm Variable Hierarchy

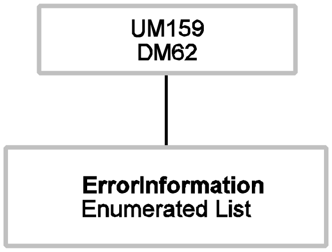


Figure 4.6-34. ErrorInformation Variable Hierarchy

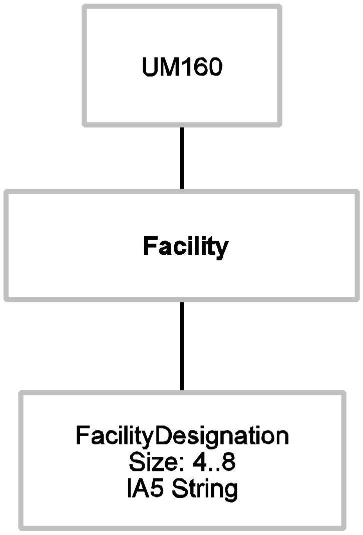


Figure 4.6-35. Facility Variable Hierarchy

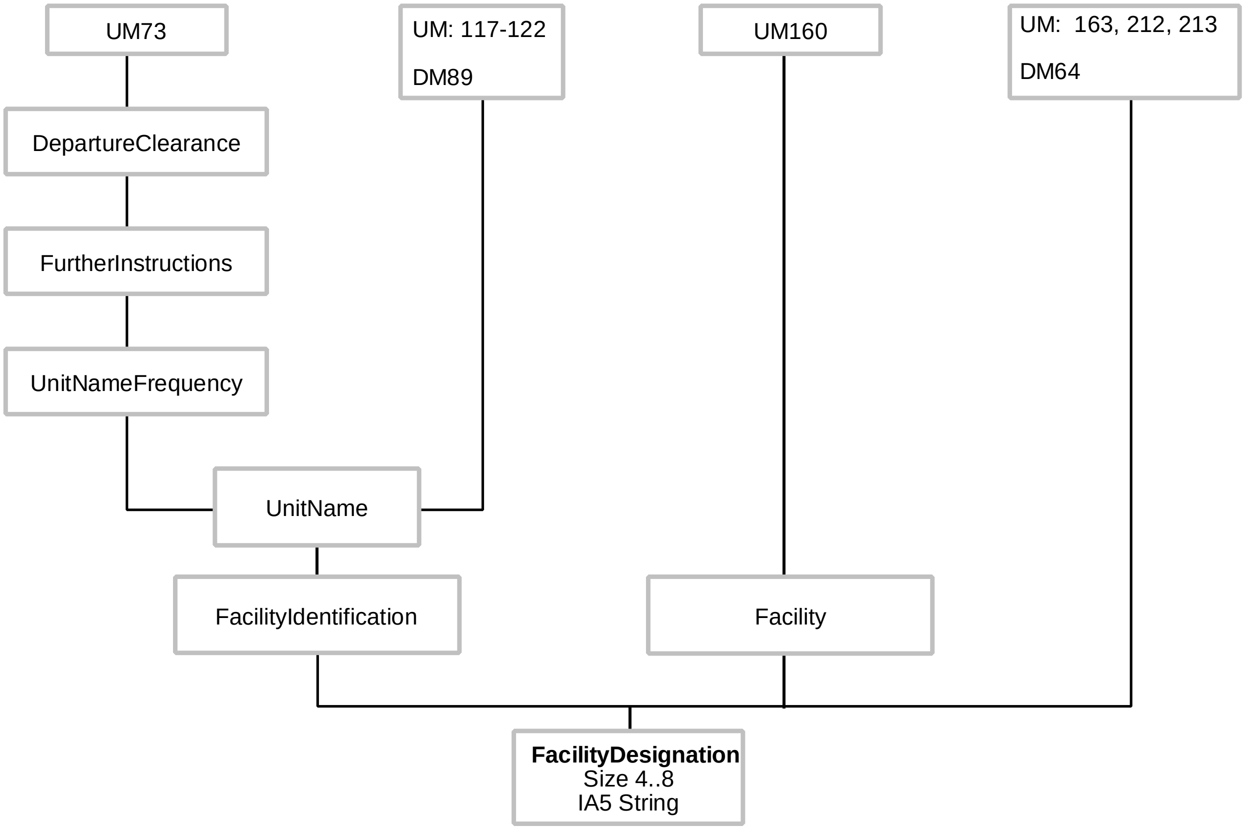


Figure 4.6-36. FacilityDesignation Variable Hierarchy

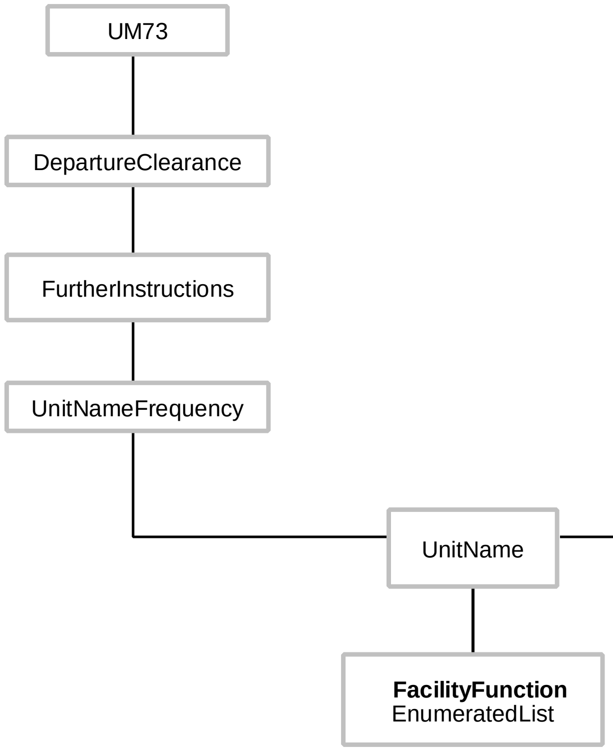


Figure 4.6-37. FacilityFunction Variable Hierarchy

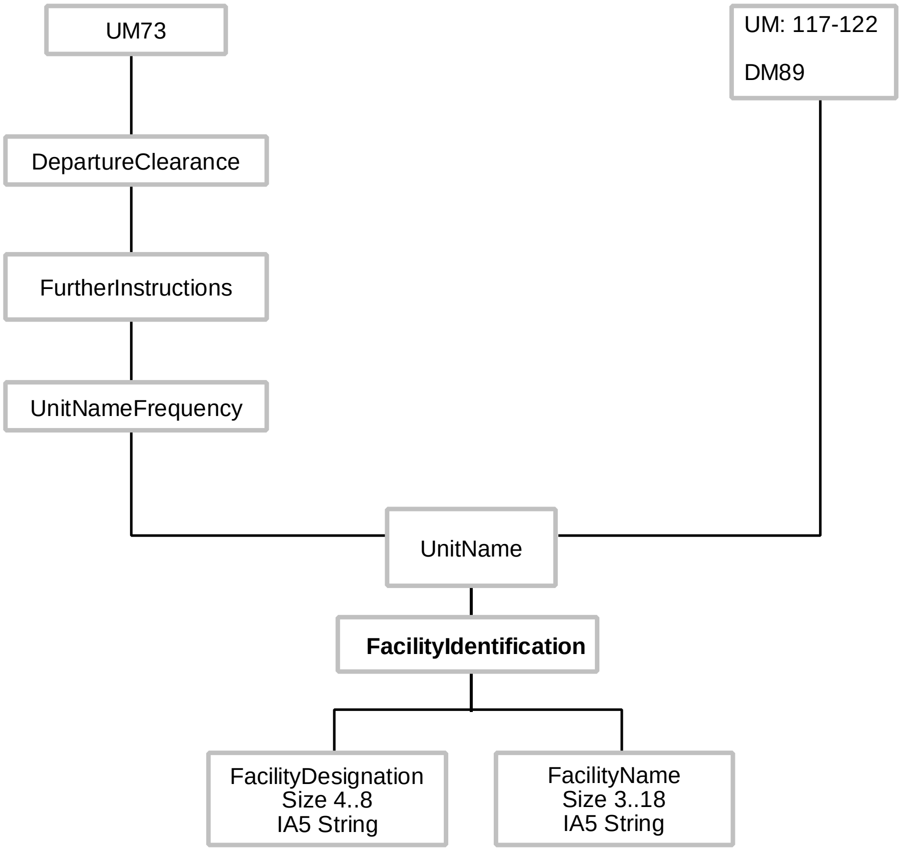


Figure 4.6-38. FacilityIdentification Variable Hierarchy

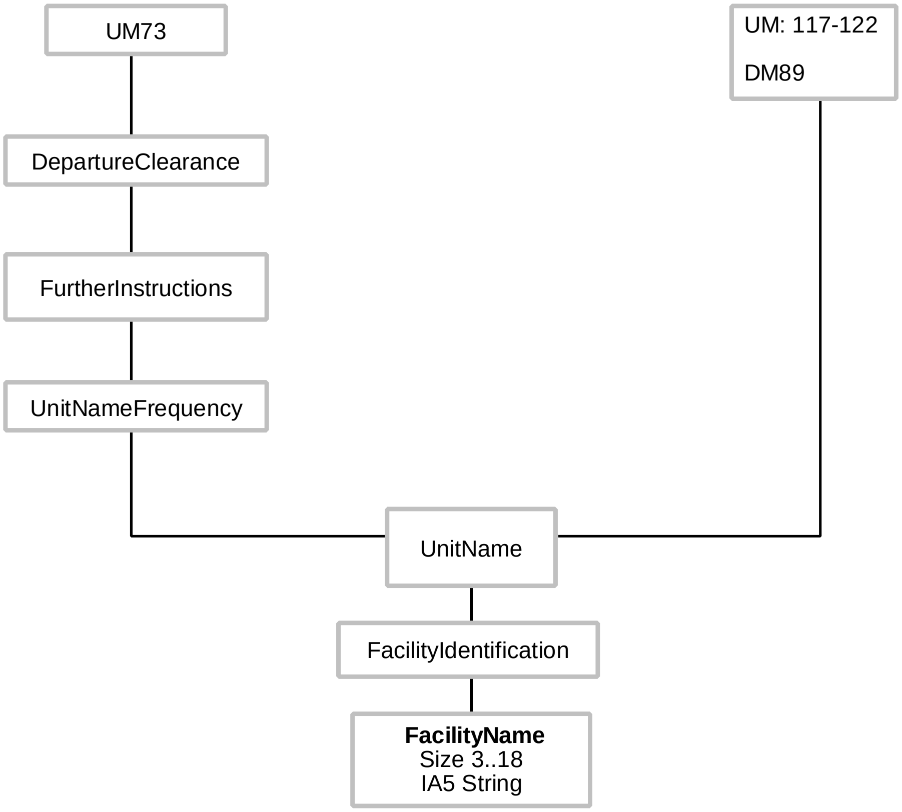


Figure 4.6-39. FacilityName Variable Hierarchy

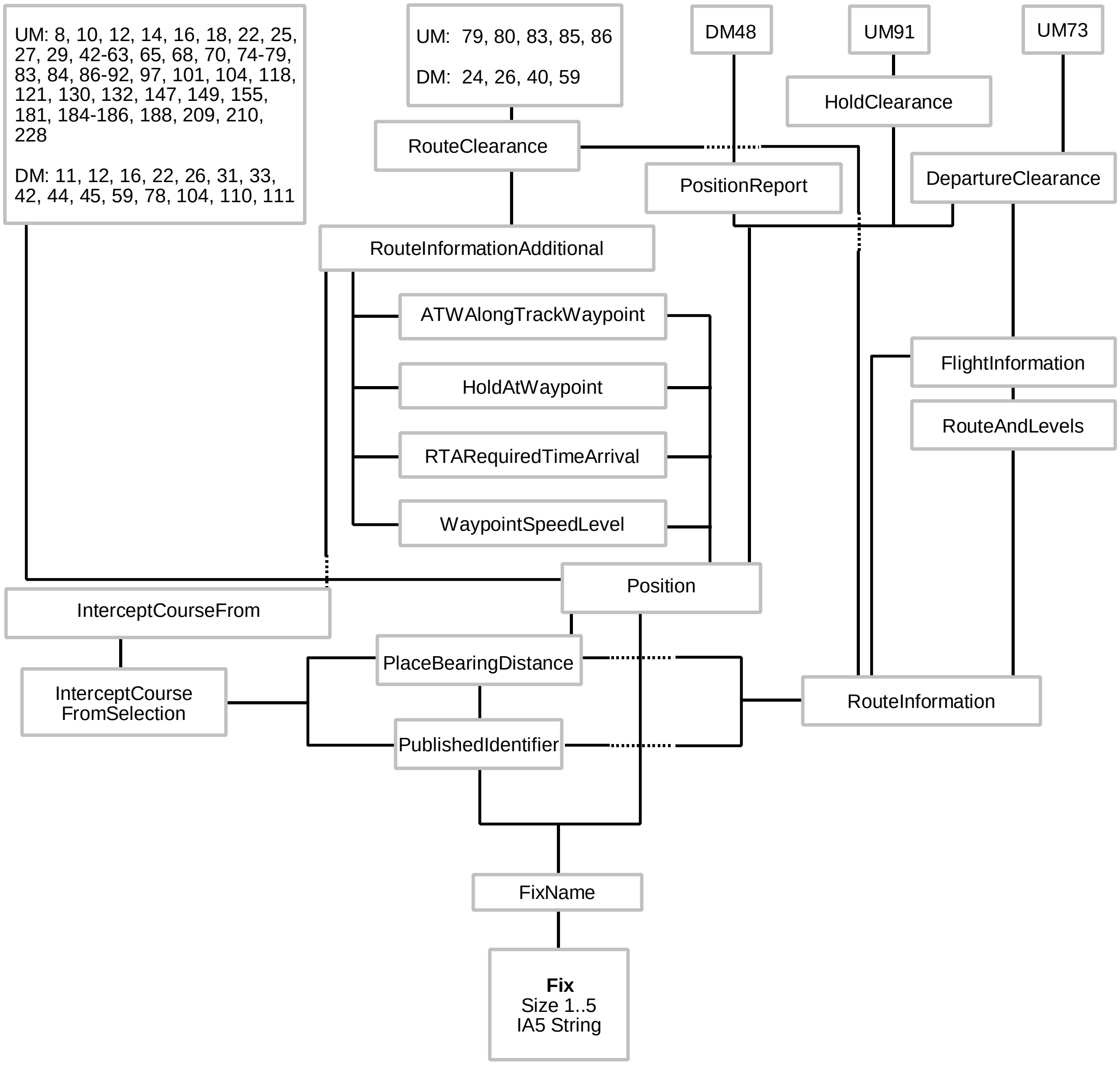


Figure 4.6-40. Fix Variable Hierarchy

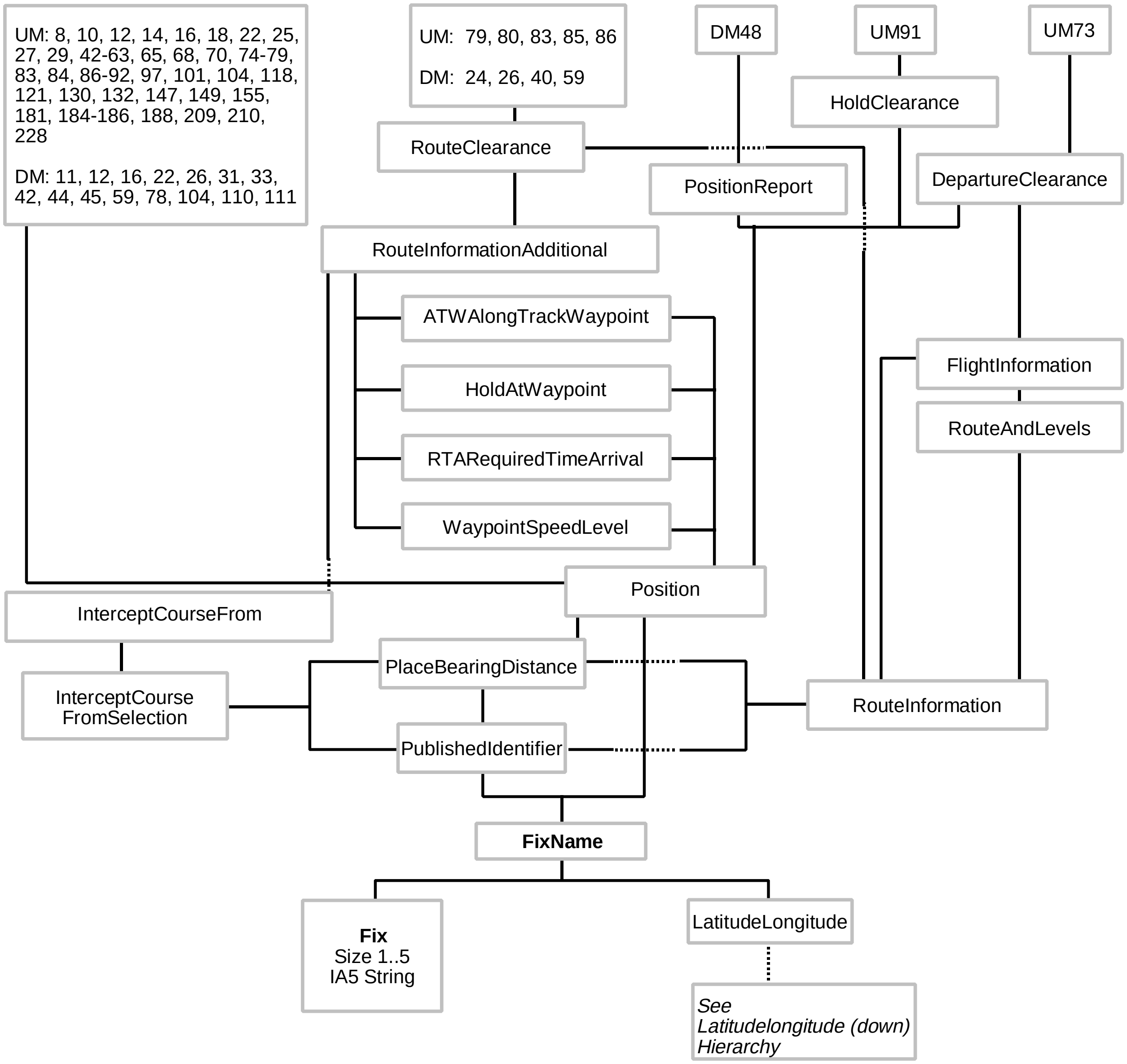


Figure 4.6-41. FixName Variable Hierarchy

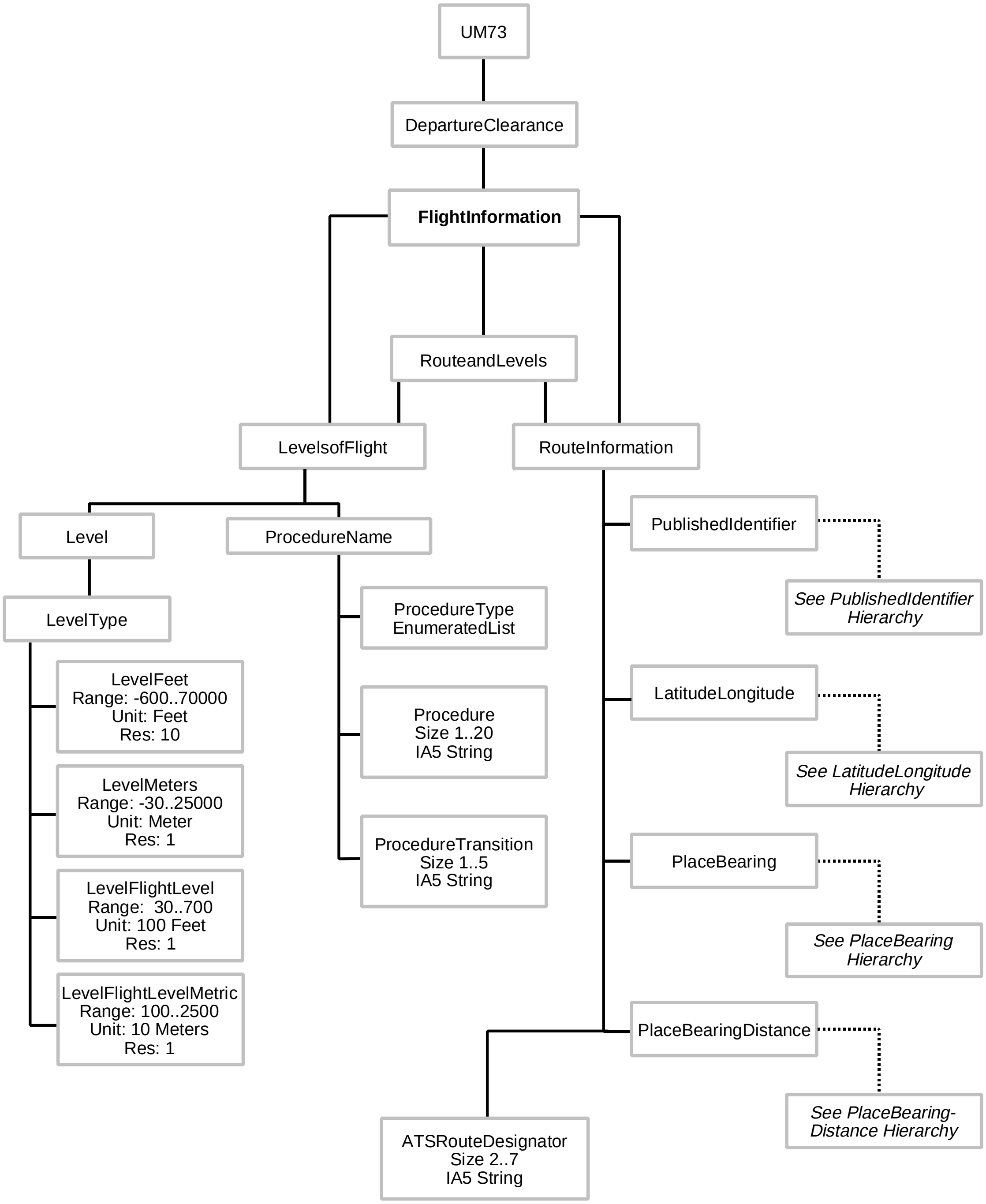

UM73
DepartureClearance
FlightInformation
RouteandLevels
LevelsofFlight
RouteInformation
Level
ProcedureName
LevelType
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
ProcedureType EnumeratedList
Procedure Size 1..20 IA5 String
ProcedureTransition Size 1..5 IA5 String
PublishedIdentifier
See PublishedIdentifier Hierarchy
LatitudeLongitude
See LatitudeLongitude Hierarchy
PlaceBearing
See PlaceBearing Hierarchy
PlaceBearingDistance
See PlaceBearing-Distance Hierarchy
ATSRouteDesignator Size 2..7 IA5 String

Figure 4.6-42. FlightInformation Variable Hierarchy

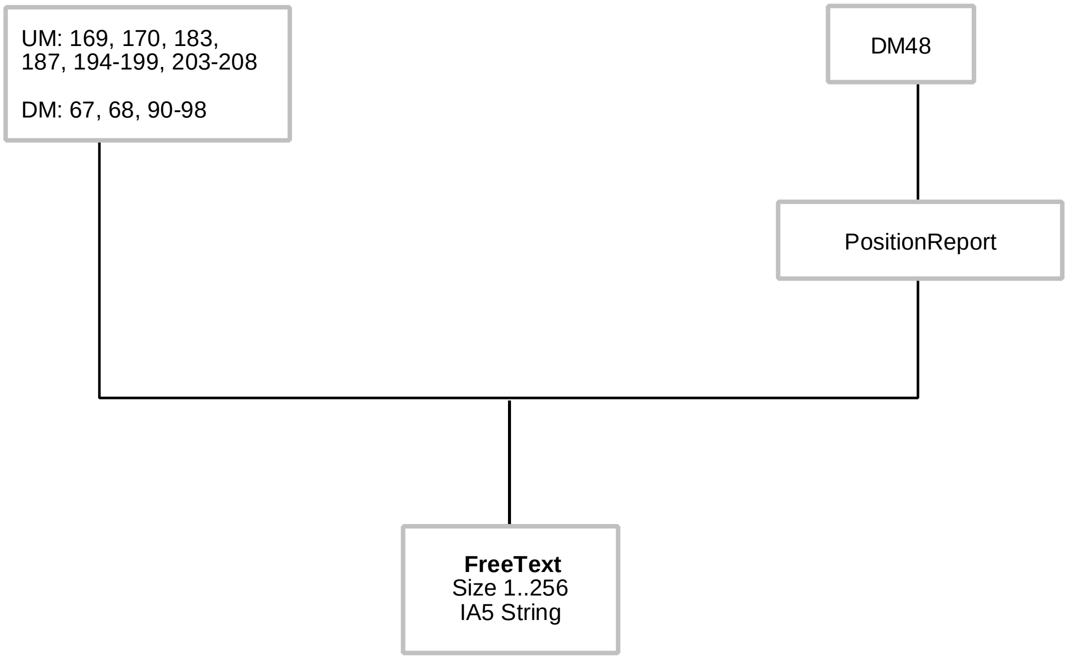


Figure 4.6-43. FreeText Variable Hierarchy

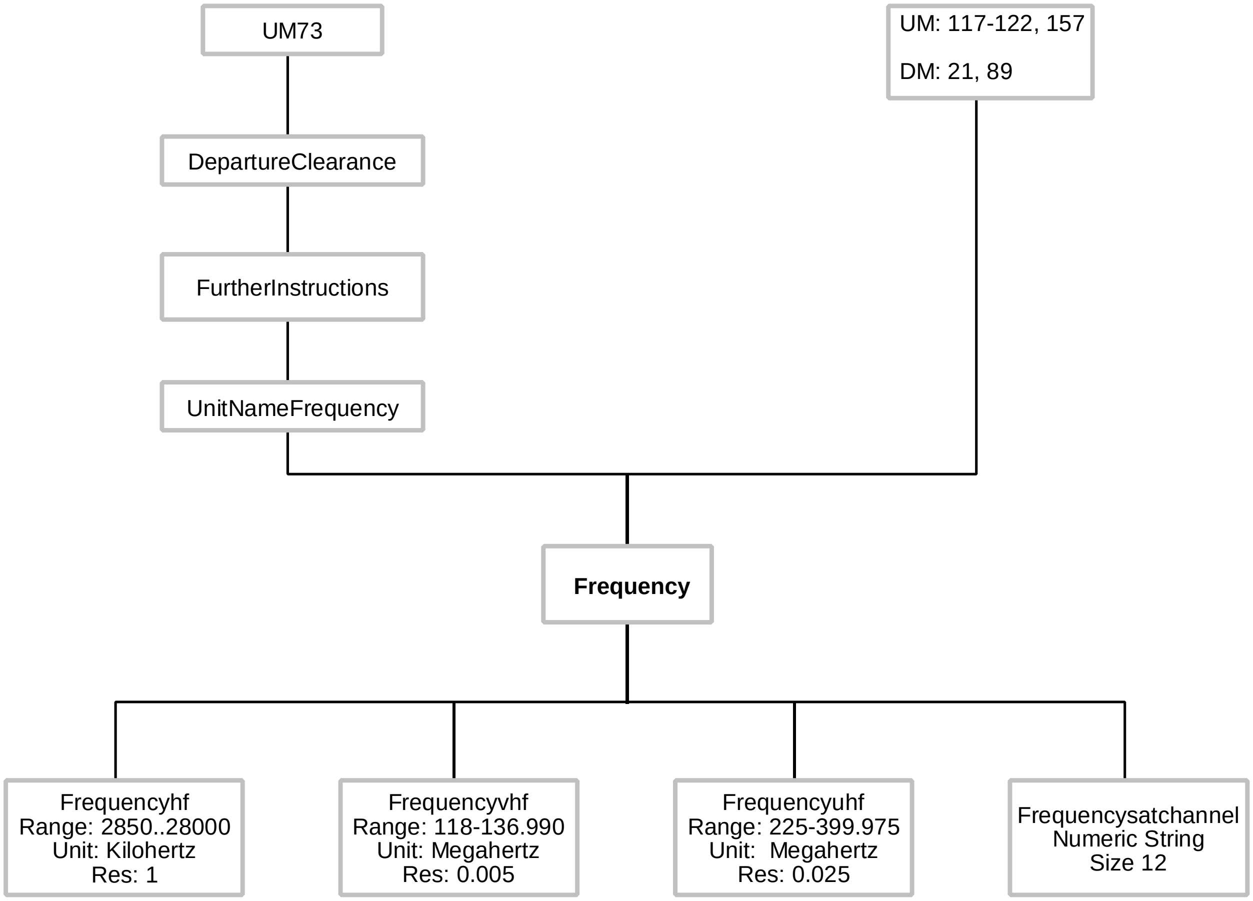


Figure 4.6-44. Frequency Variable Hierarchy

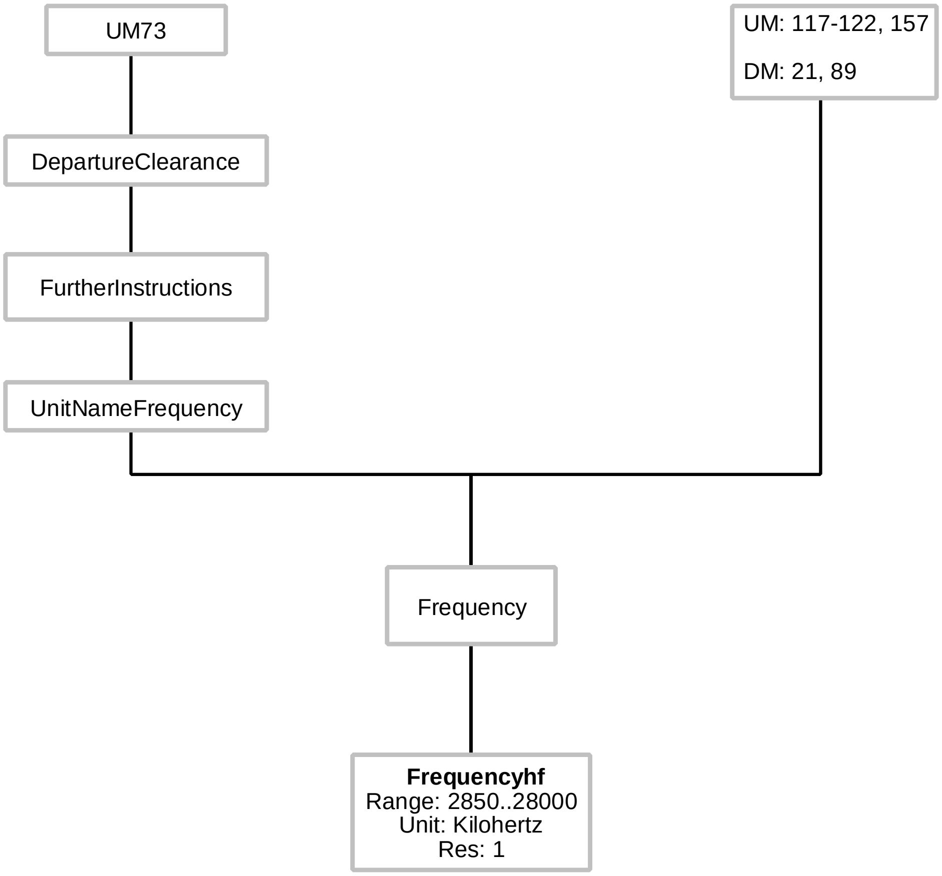


Figure 4.6-45. Frequencyhf Variable Hierarchy

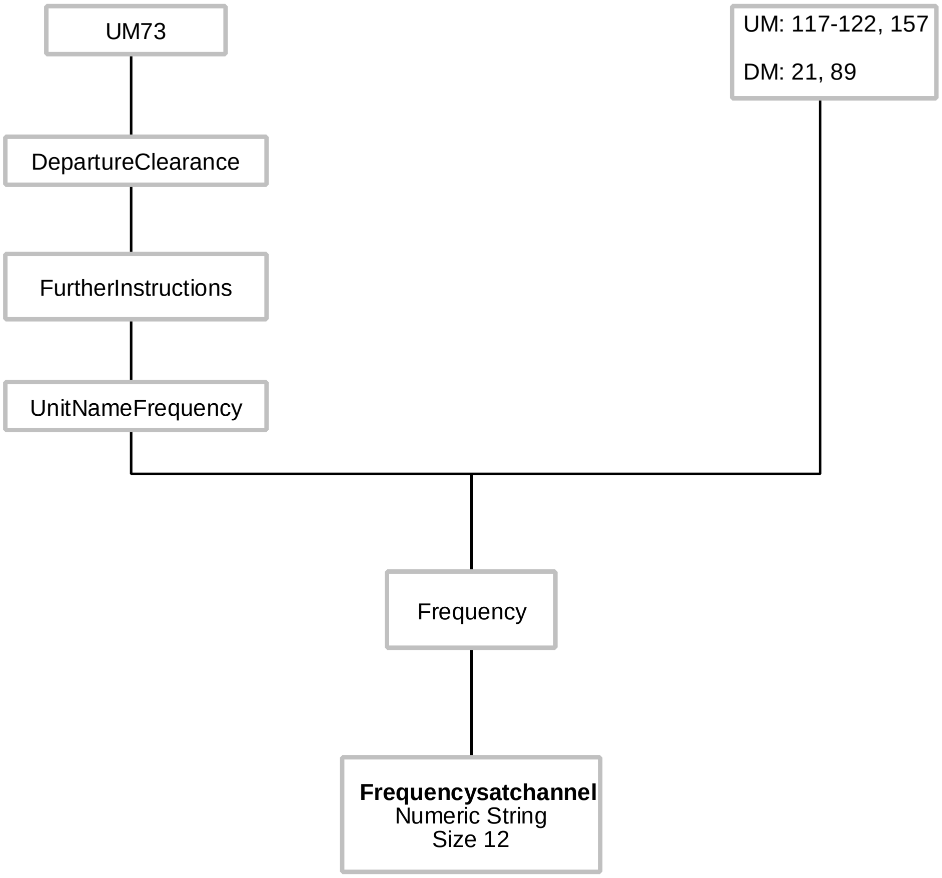


Figure 4.6-46. Frequencysatchannel Variable Hierarchy

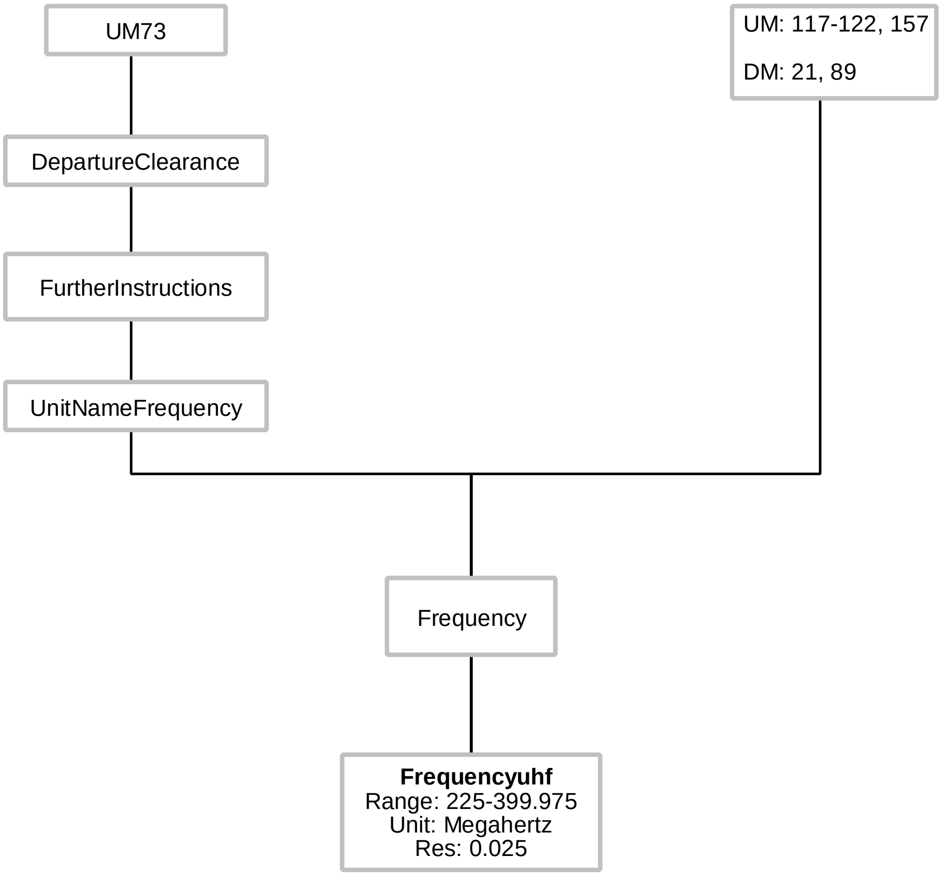


Figure 4.6-47. Frequencyuhf Variable Hierarchy

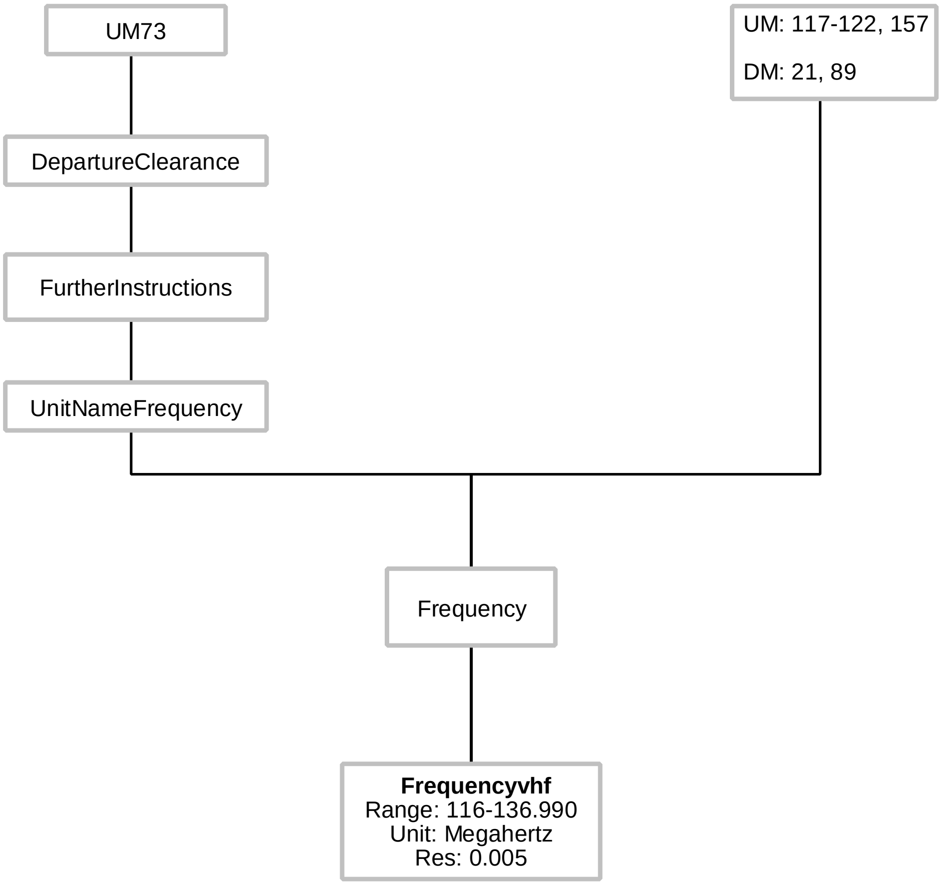


Figure 4.6-48. Frequencyvhf Variable Hierarchy

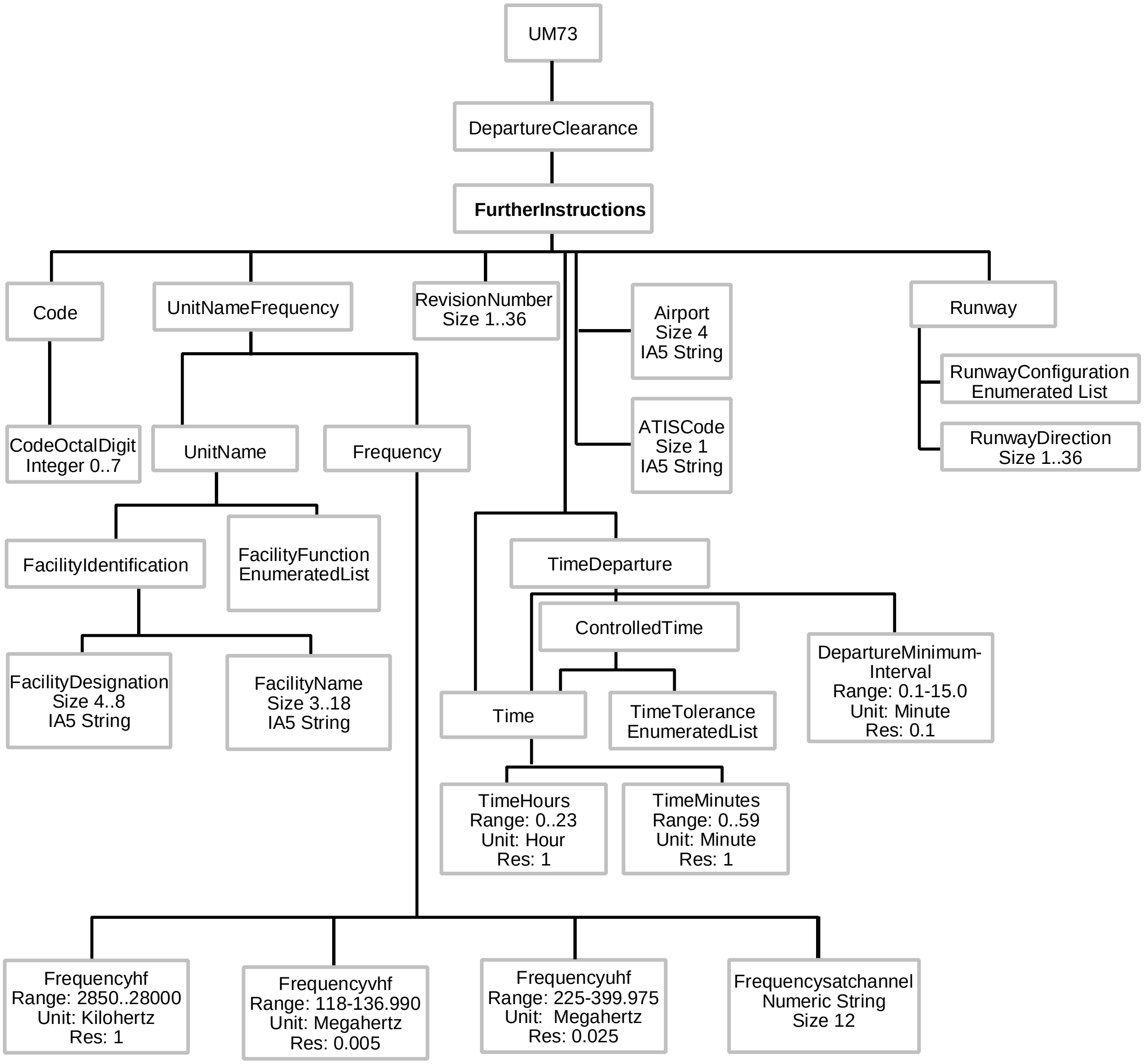

UM73
DepartureClearance
FurtherInstructions
Code
UnitNameFrequency
RevisionNumber Size 1..36
Airport Size 4 IA5 String
Runway
RunwayConfiguration Enumerated List
RunwayDirection Size 1..36
ATISCode Size 1 IA5 String
CodeOctalDigit Integer 0..7
UnitName
Frequency
FacilityIdentification
FacilityFunction EnumeratedList
TimeDeparture
ControlledTime
DepartureMinimum-Interval Range: 0.1-15.0 Unit: Minute Res: 0.1
FacilityDesignation Size 4..8 IA5 String
FacilityName Size 3..18 IA5 String
Time
TimeTolerance EnumeratedList
TimeHours Range: 0..23 Unit: Hour Res: 1
TimeMinutes Range: 0..59 Unit: Minute Res: 1
Frequencyhf Range: 2850..28000 Unit: Kilohertz Res: 1
Frequencyvhf Range: 118-136.990 Unit: Megahertz Res: 0.005
Frequencyuhf Range: 225-399.975 Unit: Megahertz Res: 0.025
Frequencysatchannel Numeric String Size 12

Figure 4.6-49. FurtherInstructions Variable Hierarchy

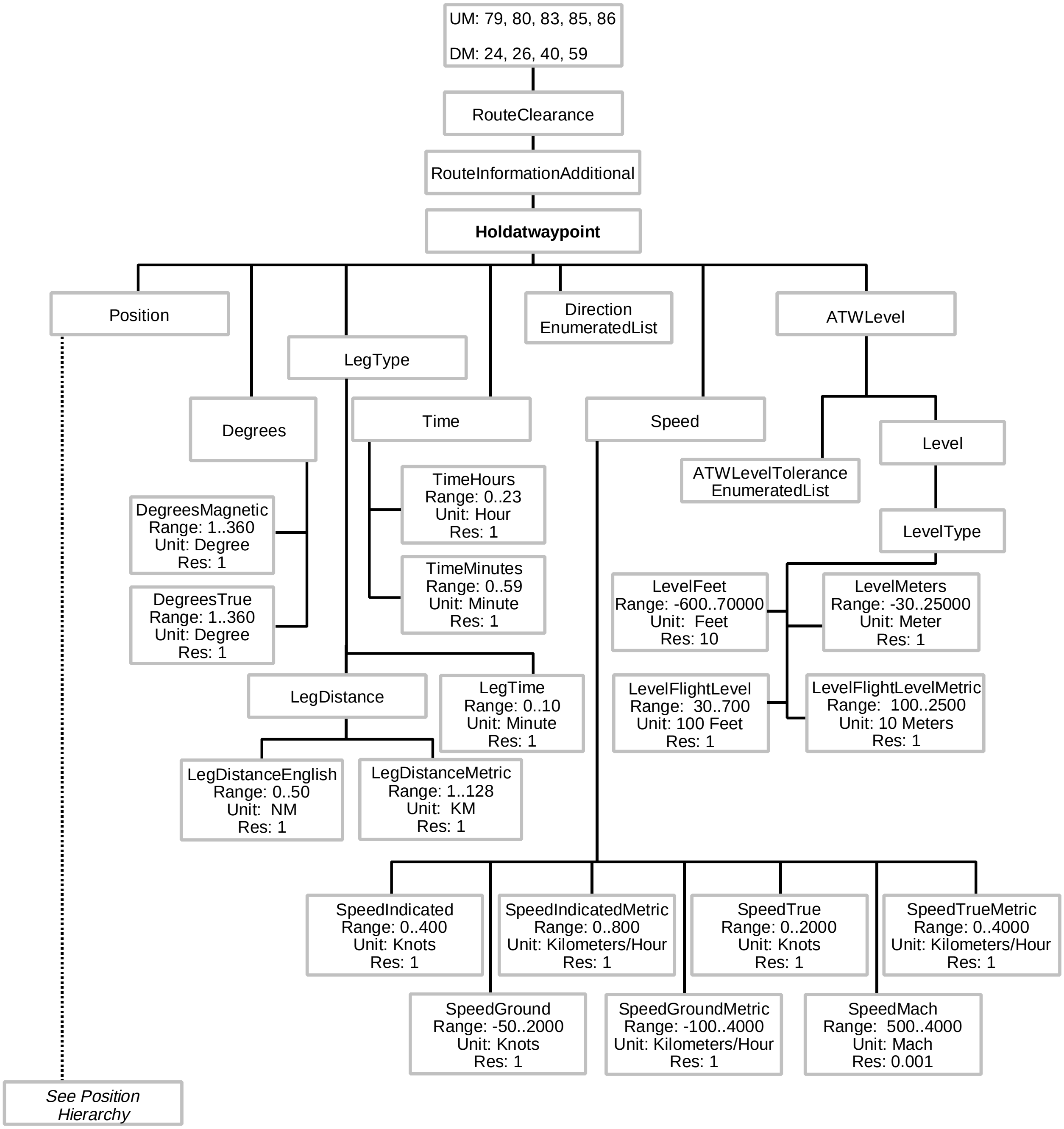

UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
RouteClearance
RouteInformationAdditional
Holdatwaypoint
Position
Direction EnumeratedList
ATWLevel
LegType
Degrees
Time
Speed
Level
ATWLevelTolerance EnumeratedList
LevelType
TimeHours Range: 0..23 Unit: Hour Res: 1
TimeMinutes Range: 0..59 Unit: Minute Res: 1
DegreesMagnetic Range: 1..360 Unit: Degree Res: 1
DegreesTrue Range: 1..360 Unit: Degree Res: 1
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
LegDistance
LegTime Range: 0..10 Unit: Minute Res: 1
LegDistanceEnglish Range: 0..50 Unit: NM Res: 1
LegDistanceMetric Range: 1..128 Unit: KM Res: 1
SpeedIndicated Range: 0..400 Unit: Knots Res: 1
SpeedIndicatedMetric Range: 0..800 Unit: Kilometers/Hour Res: 1
SpeedTrue Range: 0..2000 Unit: Knots Res: 1
SpeedTrueMetric Range: 0..4000 Unit: Kilometers/Hour Res: 1
SpeedGround Range: -50..2000 Unit: Knots Res: 1
SpeedGroundMetric Range: -100..4000 Unit: Kilometers/Hour Res: 1
SpeedMach Range: 500..4000 Unit: Mach Res: 0.001
See Position Hierarchy

Figure 4.6-50. Holdatwaypoint Variable Hierarchy

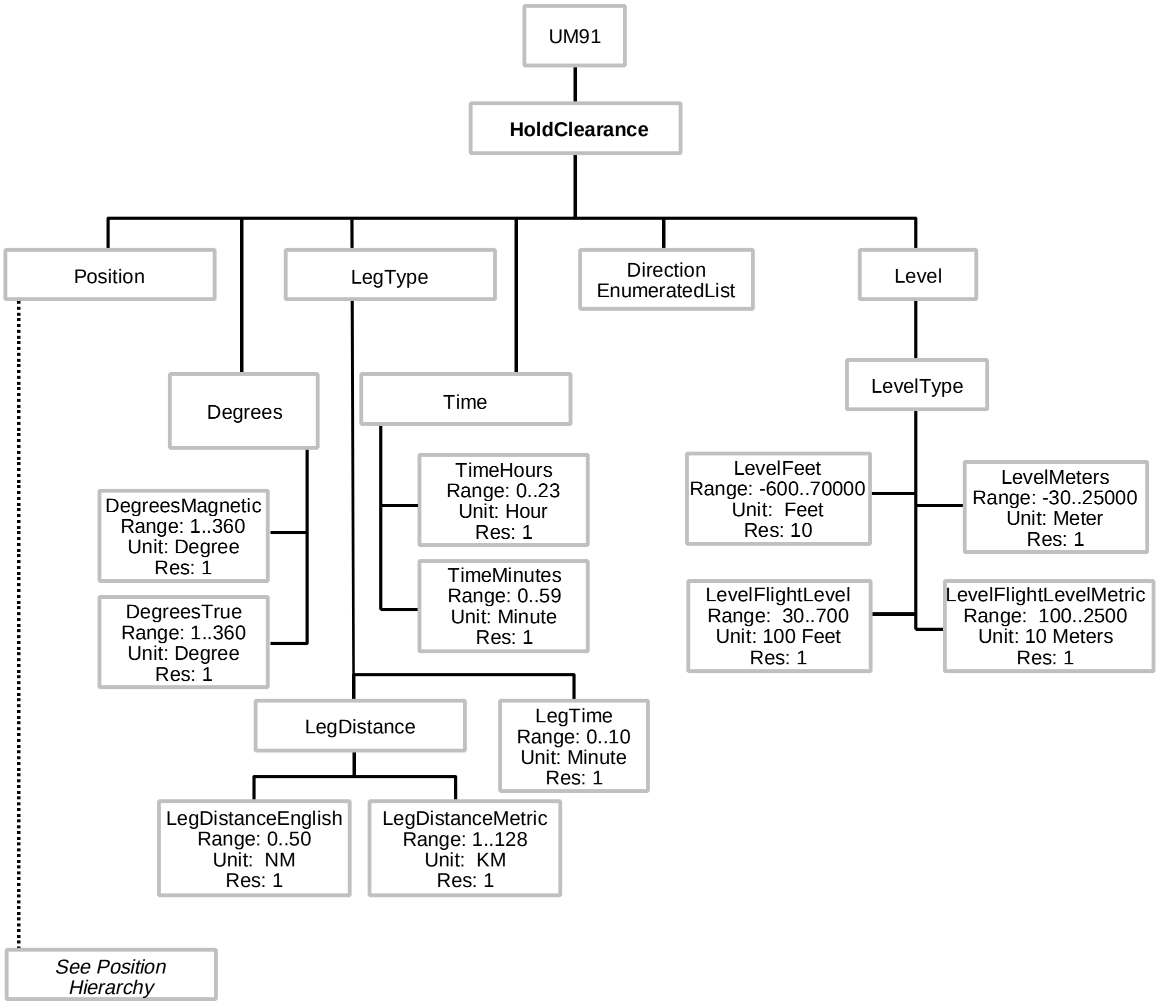

UM91
HoldClearance
Position
LegType
Direction EnumeratedList
Level
Degrees
Time
LevelType
TimeHours Range: 0..23 Unit: Hour Res: 1
TimeMinutes Range: 0..59 Unit: Minute Res: 1
DegreesMagnetic Range: 1..360 Unit: Degree Res: 1
DegreesTrue Range: 1..360 Unit: Degree Res: 1
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
LegDistance
LegTime Range: 0..10 Unit: Minute Res: 1
LegDistanceEnglish Range: 0..50 Unit: NM Res: 1
LegDistanceMetric Range: 1..128 Unit: KM Res: 1
See Position Hierarchy

Figure 4.6-51. HoldClearance Variable Hierarchy

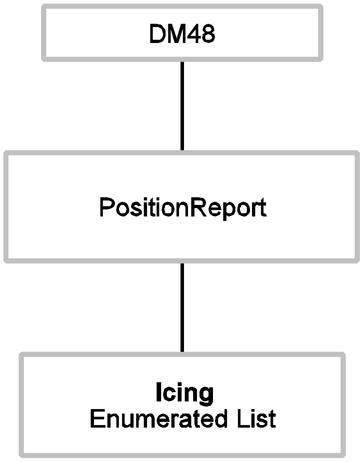


Figure 4.6-52. Icing Variable Hierarchy

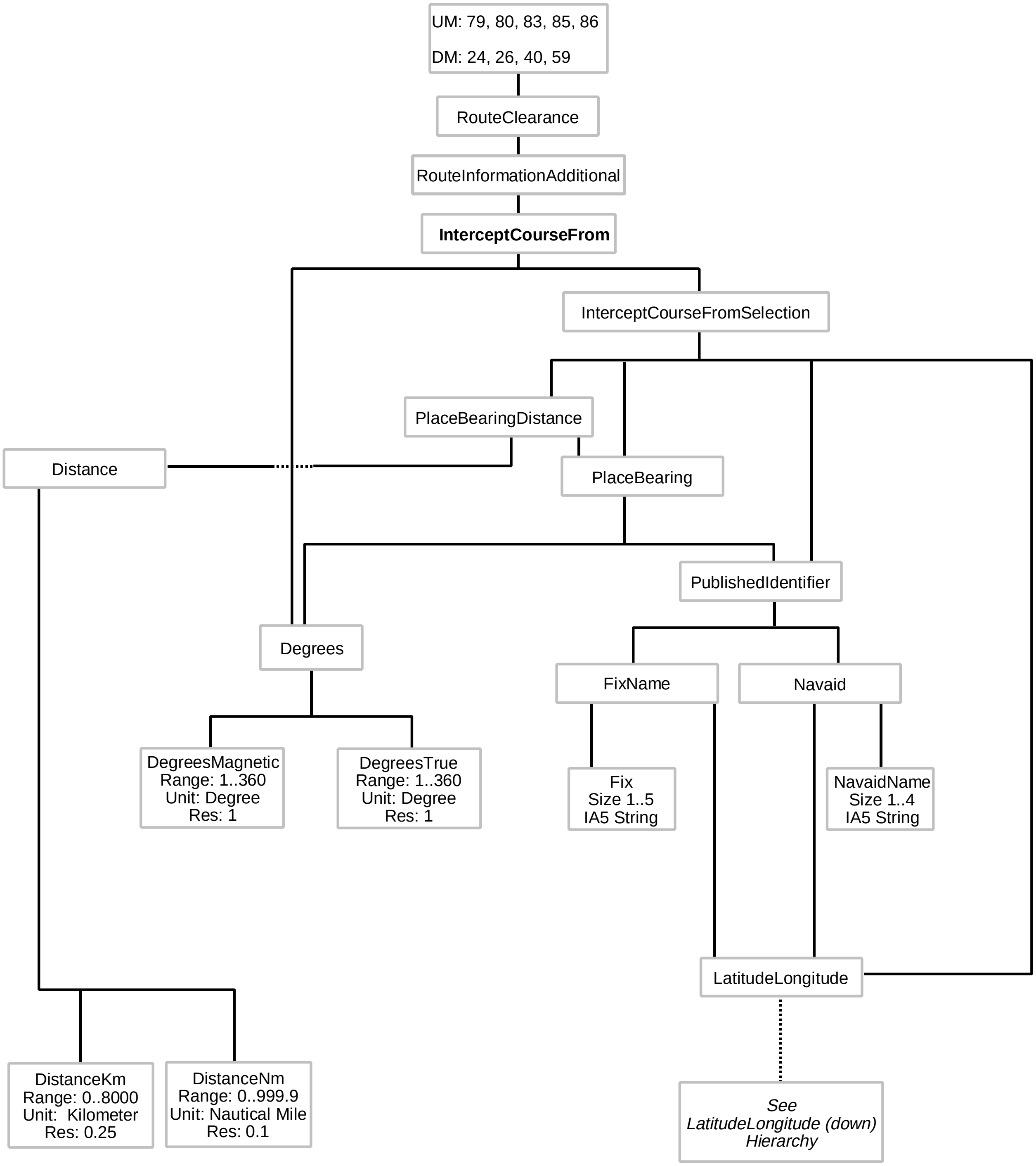

UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
RouteClearance
RouteInformationAdditional
InterceptCourseFrom
InterceptCourseFromSelection
PlaceBearingDistance
Distance
PlaceBearing
PublishedIdentifier
Degrees
FixName
Navaid
DegreesMagnetic Range: 1..360 Unit: Degree Res: 1
DegreesTrue Range: 1..360 Unit: Degree Res: 1
Fix Size 1..5 IA5 String
NavaidName Size 1..4 IA5 String
LatitudeLongitude
DistanceKm Range: 0..8000 Unit: Kilometer Res: 0.25
DistanceNm Range: 0..999.9 Unit: Nautical Mile Res: 0.1
See LatitudeLongitude (down) Hierarchy

Figure 4.6-53. InterceptCourseFrom Variable Hierarchy

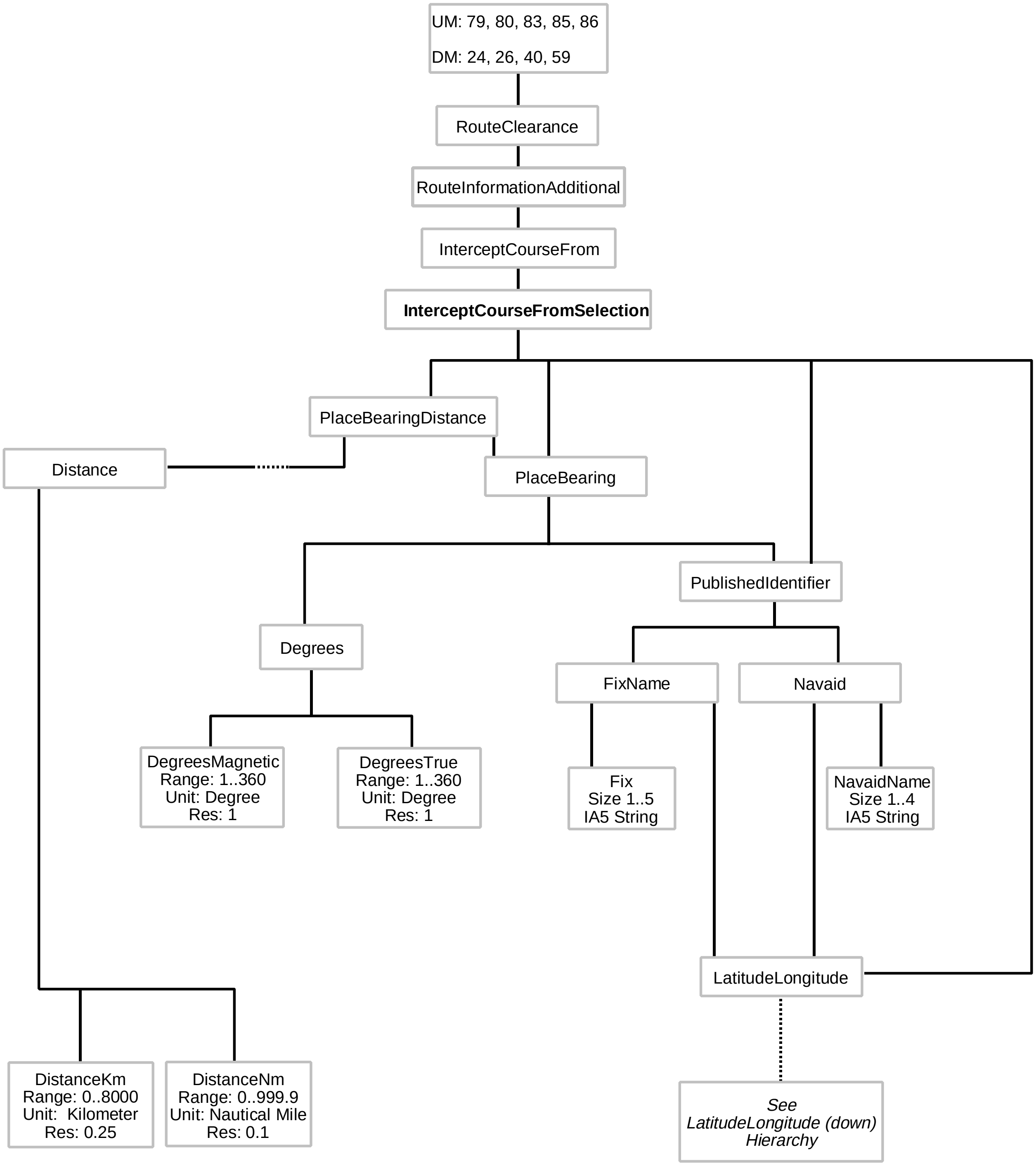


Figure 4.6-54. InterceptCourseFromSelection Variable Hierarchy

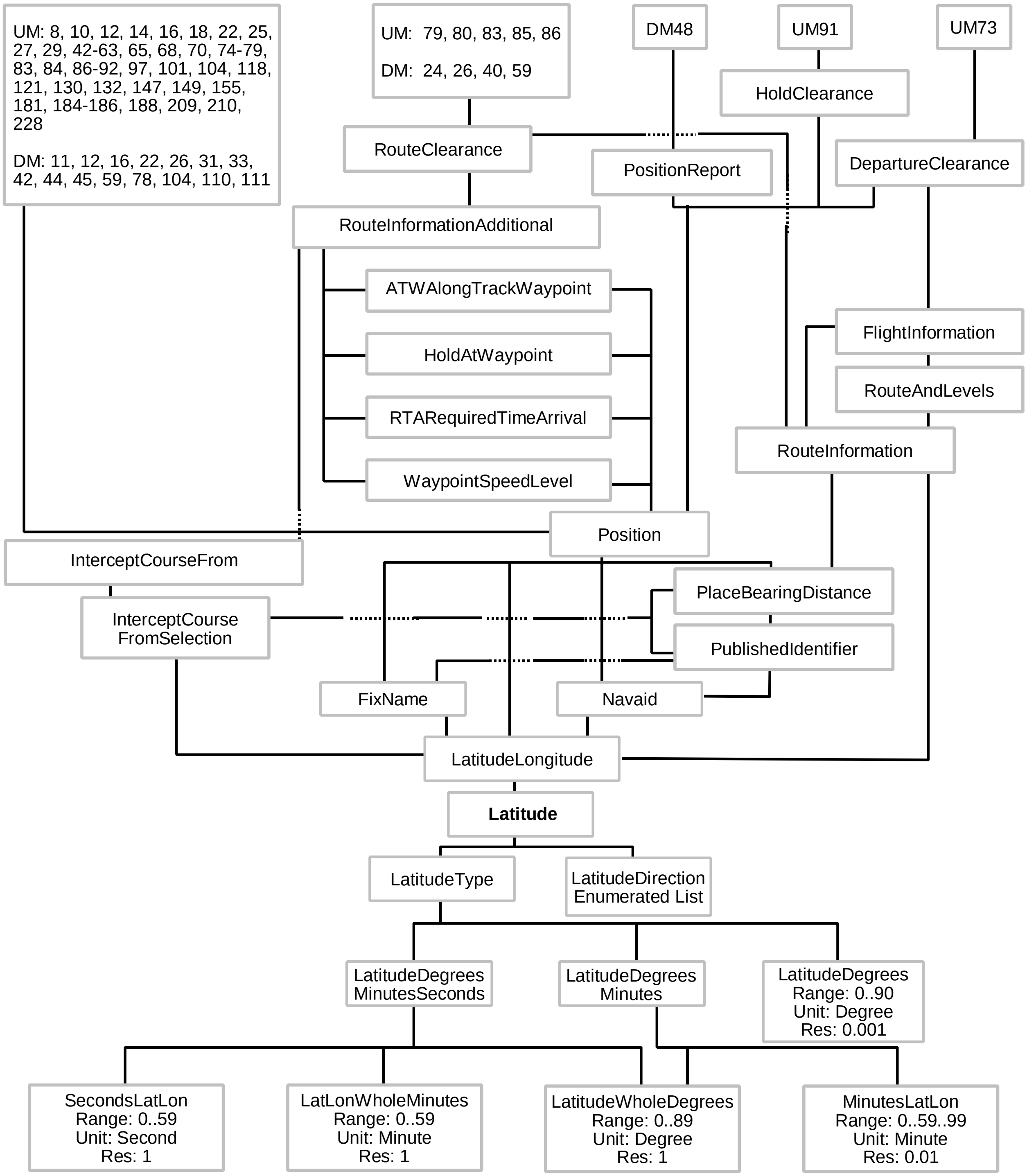

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 74-79, 83, 84, 86-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 26, 31, 33, 42, 44, 45, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
UM91
UM73
HoldClearance
RouteClearance
PositionReport
DepartureClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldAtWaypoint
RTARequiredTimeArrival
WaypointSpeedLevel
FlightInformation
RouteAndLevels
RouteInformation
Position
InterceptCourseFrom
PlaceBearingDistance
InterceptCourse FromSelection
PublishedIdentifier
FixName
Navaid
LatitudeLongitude
Latitude
LatitudeType
LatitudeDirection Enumerated List
LatitudeDegrees MinutesSeconds
LatitudeDegrees Minutes
LatitudeDegrees Range: 0..90 Unit: Degree Res: 0.001
SecondsLatLon Range: 0..59 Unit: Second Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
LatitudeWholeDegrees Range: 0..89 Unit: Degree Res: 1
MinutesLatLon Range: 0..59..99 Unit: Minute Res: 0.01

Figure 4.6-55. Latitude Variable Hierarchy

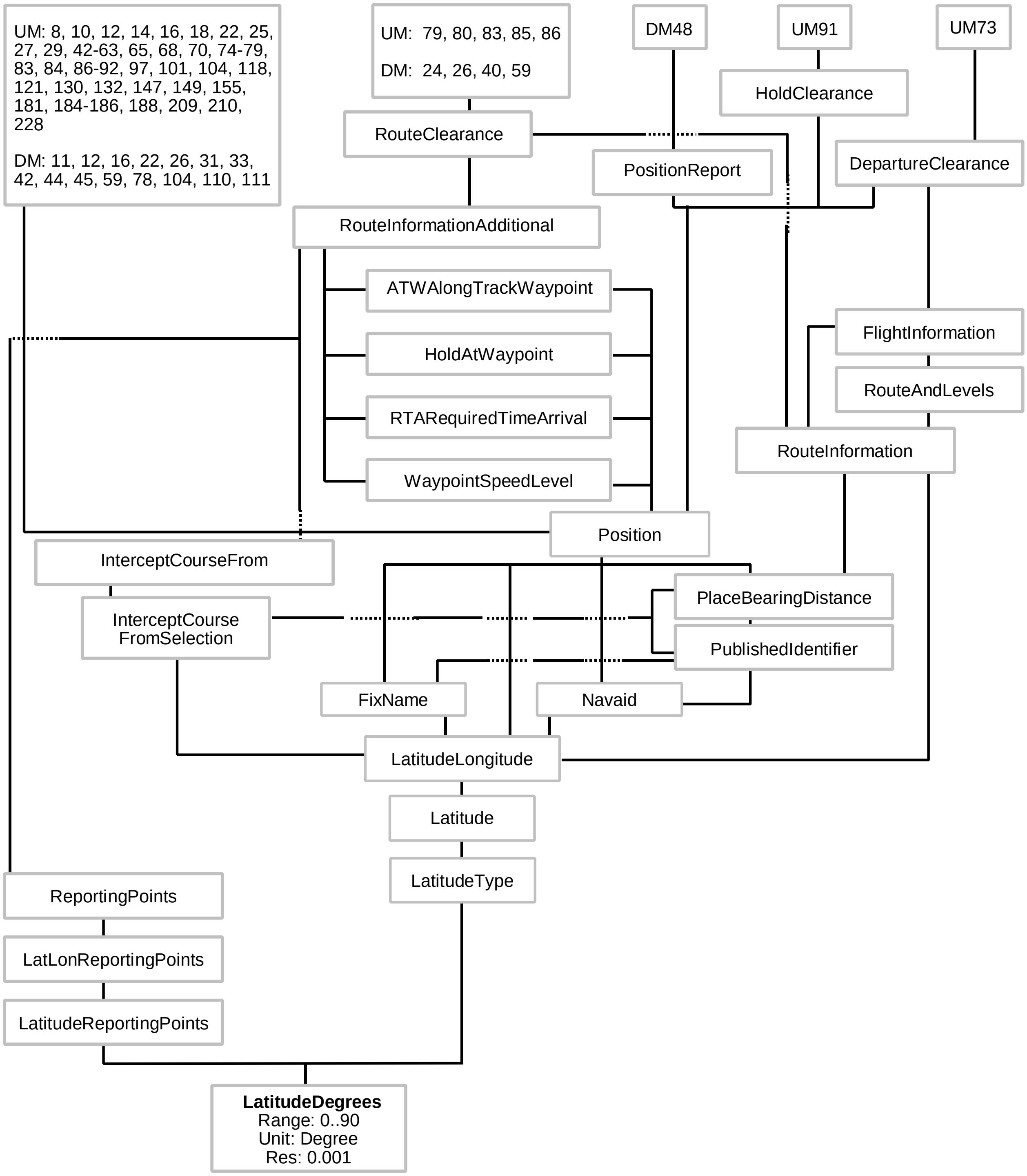


Figure 4.6-56. LatitudeDegrees Variable Hierarchy

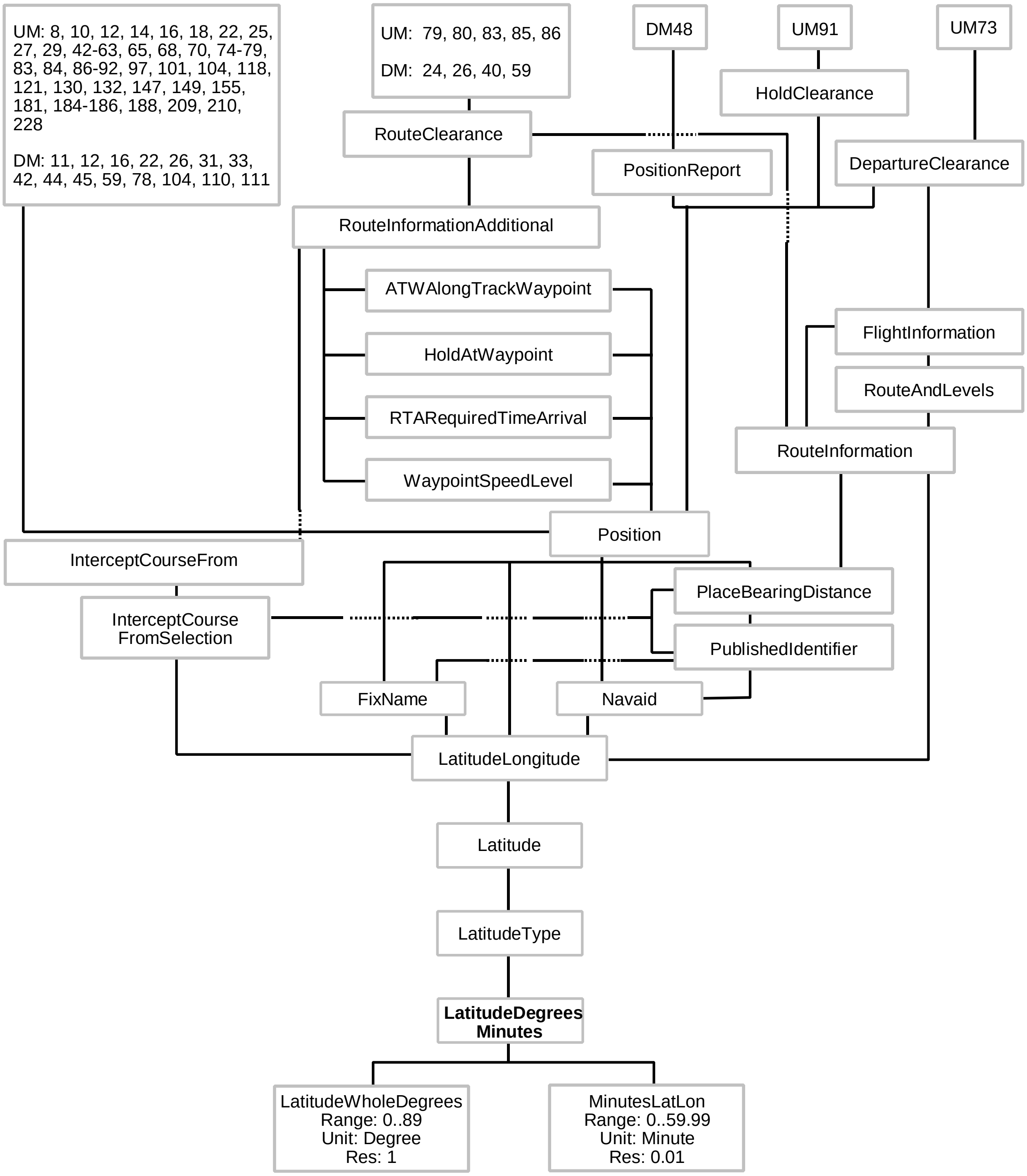


Figure 4.6-57. LatitudeDegreesMinutes Variable Hierarchy

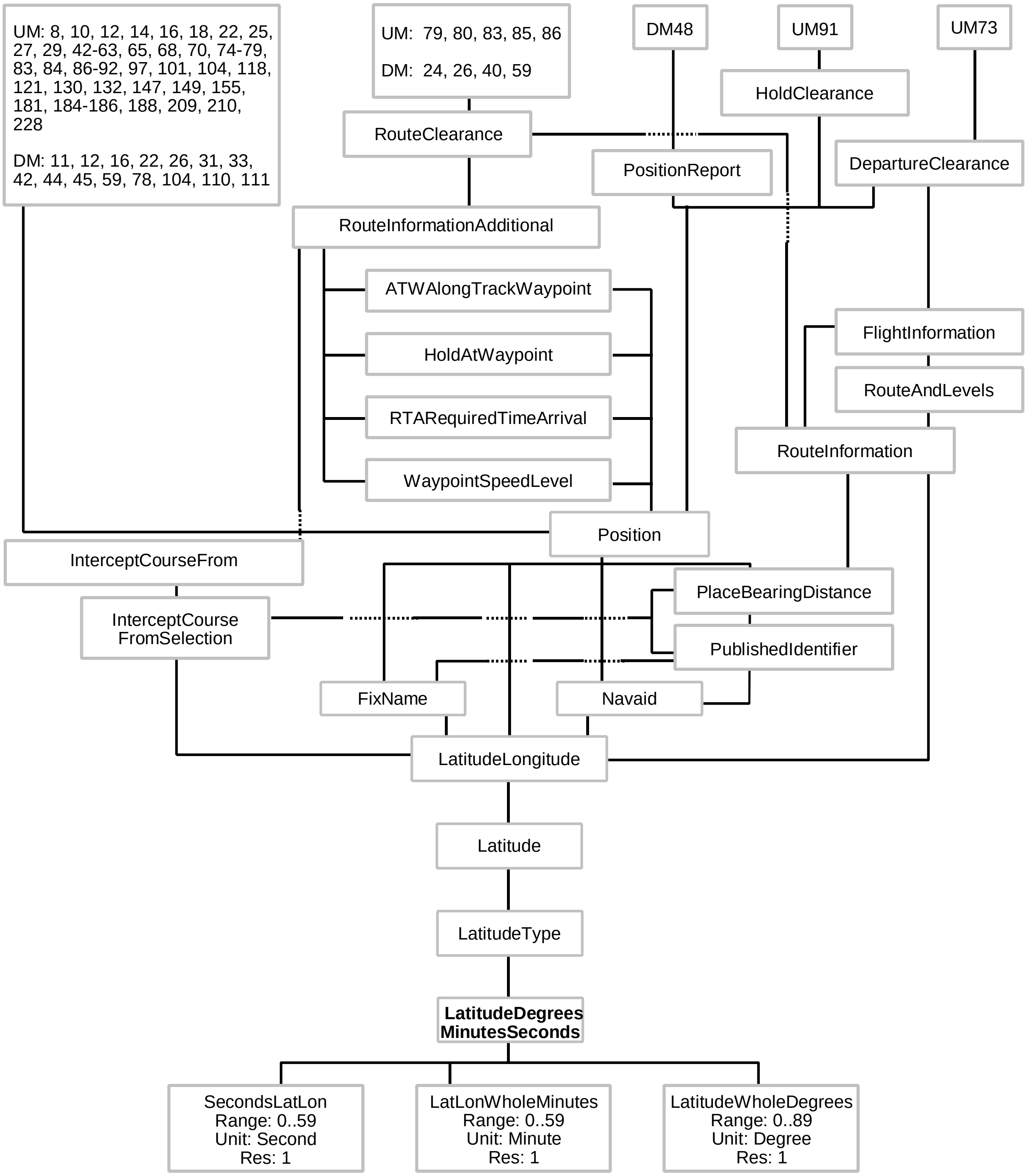

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 74-79, 83, 84, 86-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 26, 31, 33, 42, 44, 45, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
UM91
UM73
HoldClearance
RouteClearance
PositionReport
DepartureClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldAtWaypoint
RTARequiredTimeArrival
WaypointSpeedLevel
FlightInformation
RouteAndLevels
RouteInformation
Position
InterceptCourseFrom
InterceptCourse FromSelection
PlaceBearingDistance
PublishedIdentifier
FixName
Navaid
LatitudeLongitude
Latitude
LatitudeType
LatitudeDegrees MinutesSeconds
SecondsLatLon Range: 0..59 Unit: Second Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
LatitudeWholeDegrees Range: 0..89 Unit: Degree Res: 1

Figure 4.6-58. LatitudeDegreesMinutesSeconds Variable Hierarchy

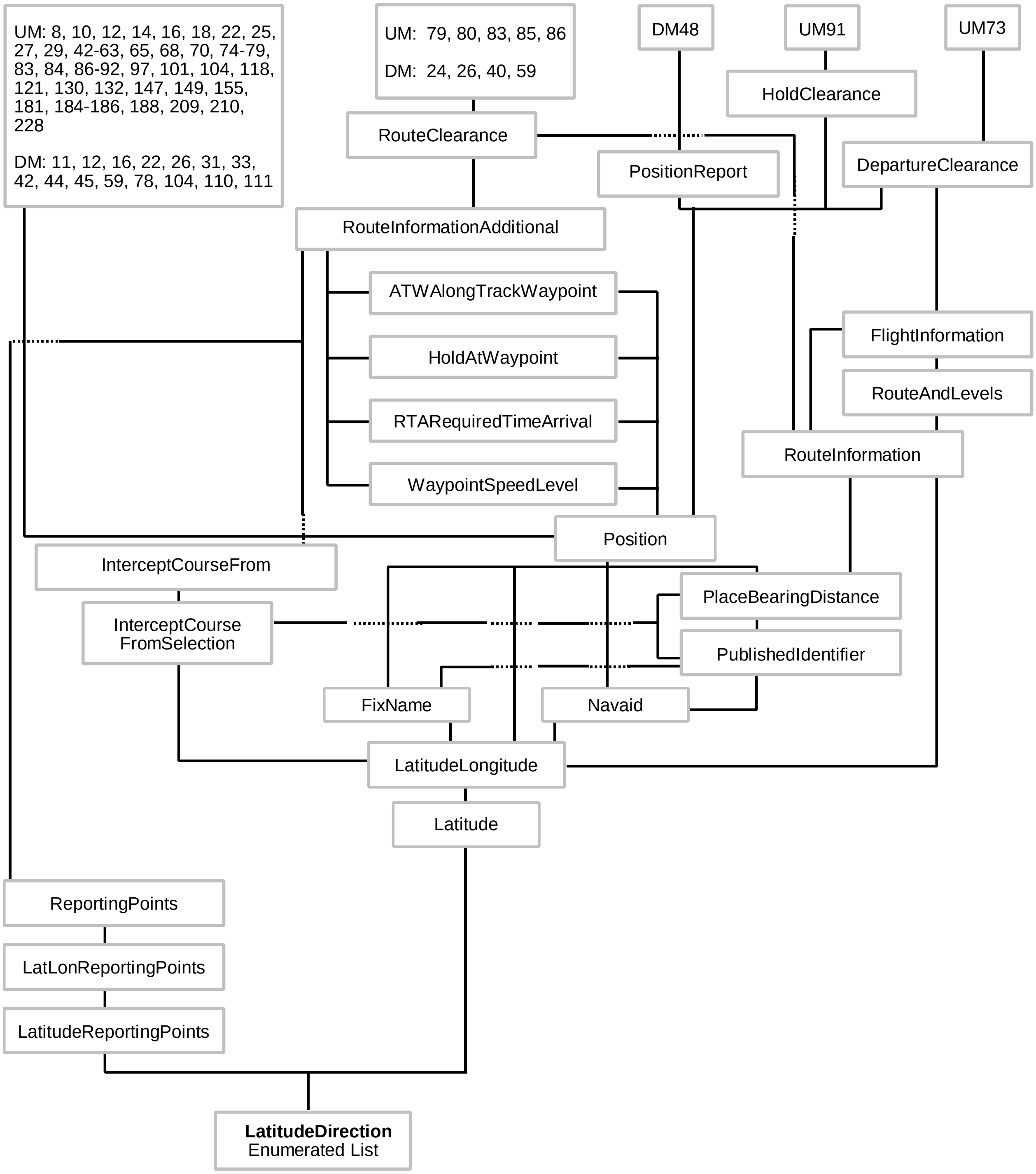


Figure 4.6-59. LatitudeDirection Variable Hierarchy

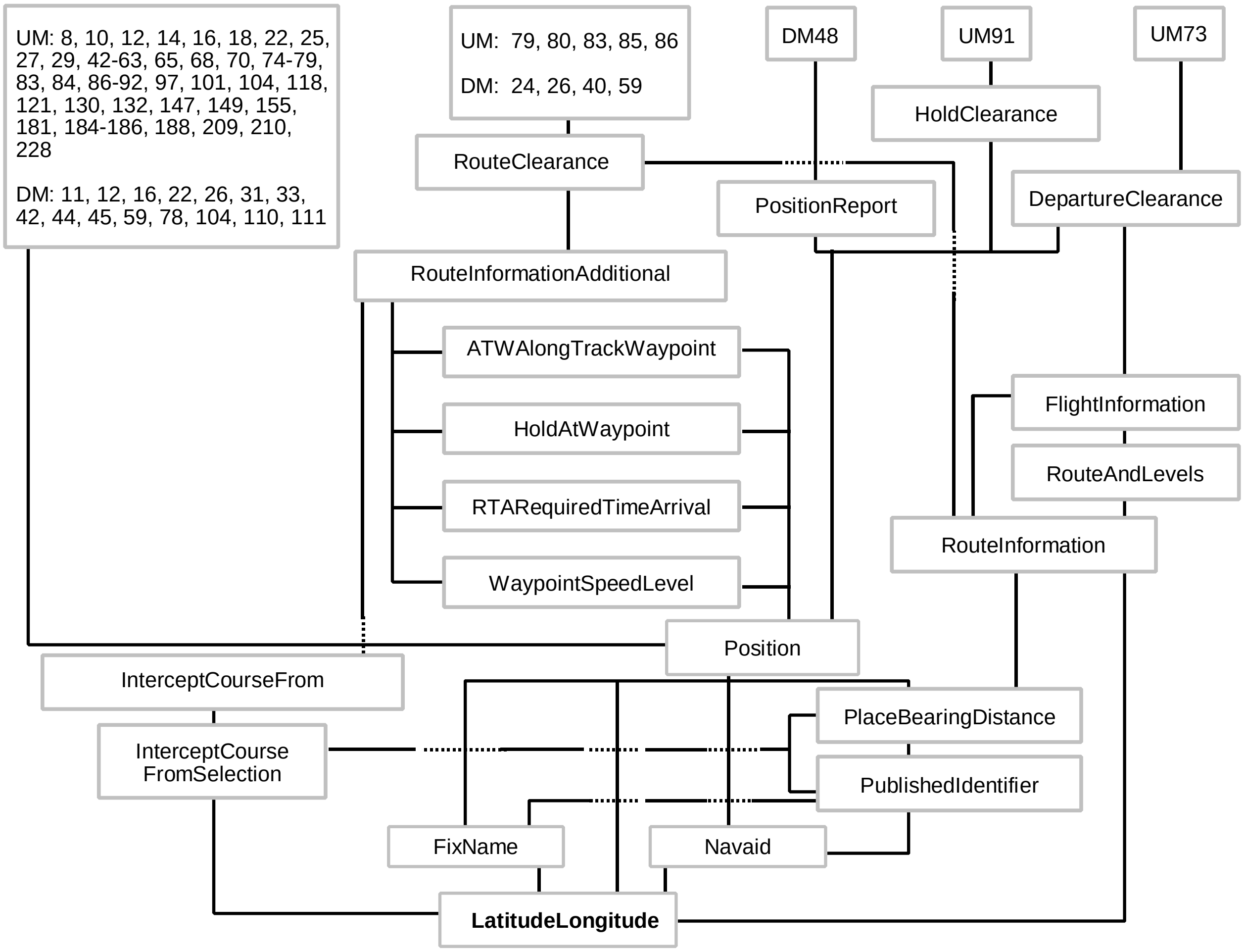


Figure 4.6-60. LatitudeLongitude (Up) Variable Hierarchy

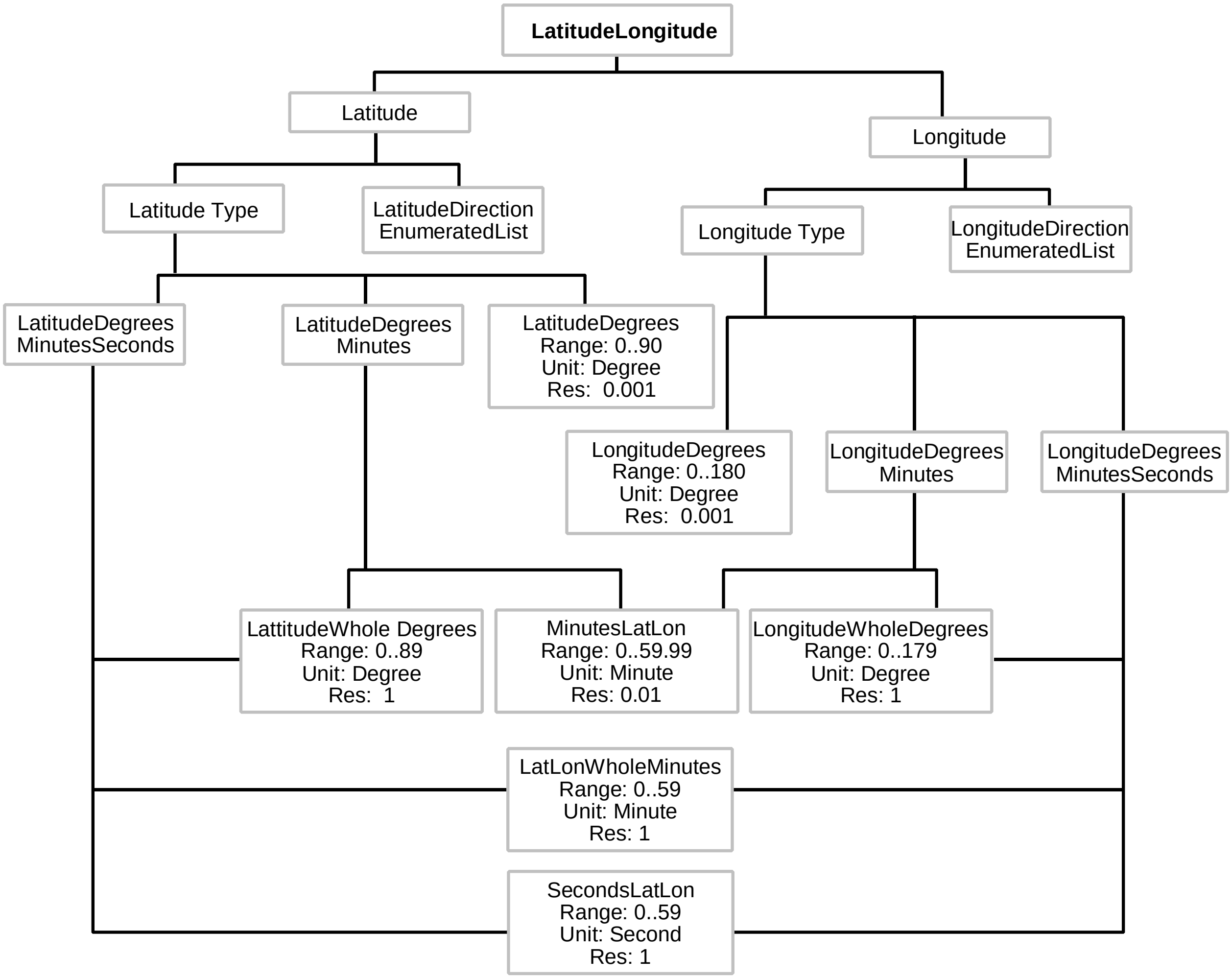

LatitudeLongitude
Latitude
Longitude
Latitude Type
LatitudeDirection EnumeratedList
Longitude Type
LongitudeDirection EnumeratedList
LatitudeDegrees MinutesSeconds
LatitudeDegrees Minutes
LatitudeDegrees Range: 0..90 Unit: Degree Res: 0.001
LongitudeDegrees Range: 0..180 Unit: Degree Res: 0.001
LongitudeDegrees Minutes
LongitudeDegrees MinutesSeconds
LattitudeWhole Degrees Range: 0..89 Unit: Degree Res: 1
MinutesLatLon Range: 0..59.99 Unit: Minute Res: 0.01
LongitudeWholeDegrees Range: 0..179 Unit: Degree Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
SecondsLatLon Range: 0..59 Unit: Second Res: 1

Figure 4.6-61. LatitudeLongitude (Down) Variable Hierarchy

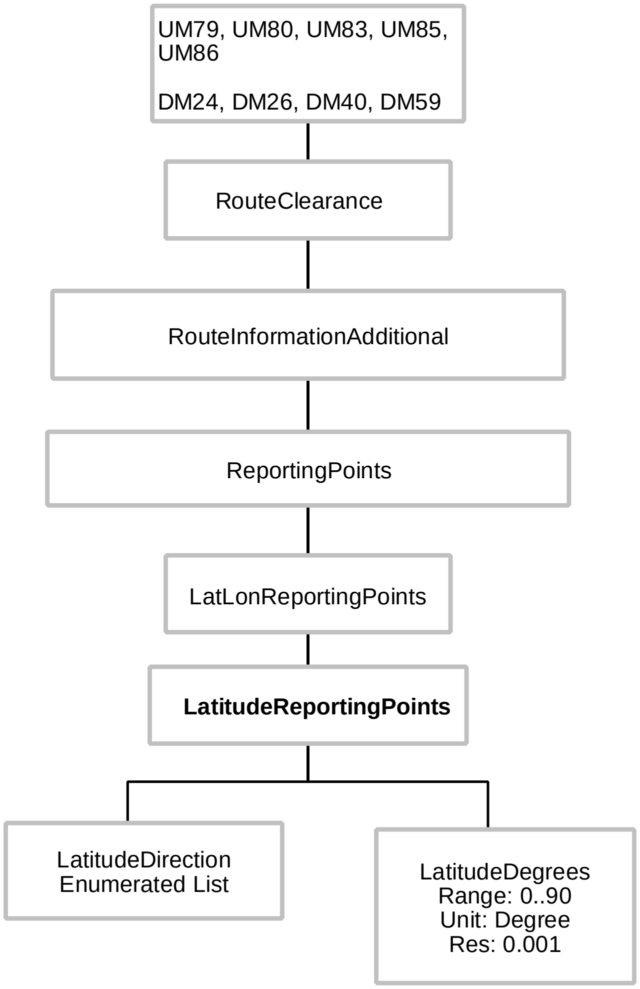

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
ReportingPoints
LatLonReportingPoints
LatitudeReportingPoints
LatitudeDirection Enumerated List
LatitudeDegrees Range: 0..90 Unit: Degree Res: 0.001

Figure 4.6-62. LatitudeReportingPoints Variable Hierarchy

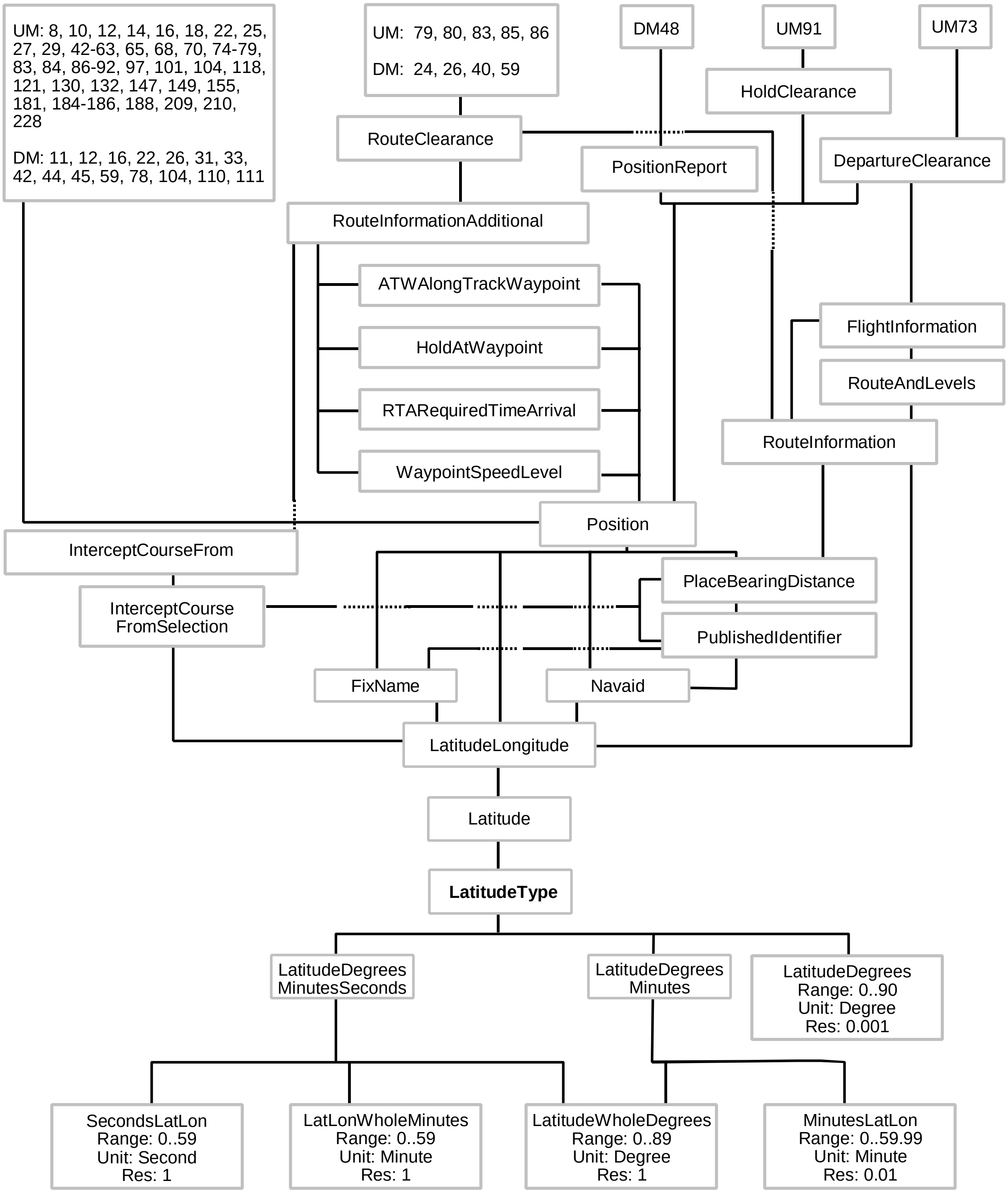

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 74-79, 83, 84, 86-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 26, 31, 33, 42, 44, 45, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
UM91
UM73
HoldClearance
RouteClearance
PositionReport
DepartureClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldAtWaypoint
RTARequiredTimeArrival
WaypointSpeedLevel
FlightInformation
RouteAndLevels
RouteInformation
Position
InterceptCourseFrom
InterceptCourse FromSelection
PlaceBearingDistance
PublishedIdentifier
FixName
Navaid
LatitudeLongitude
Latitude
LatitudeType
LatitudeDegrees MinutesSeconds
LatitudeDegrees Minutes
LatitudeDegrees Range: 0..90 Unit: Degree Res: 0.001
SecondsLatLon Range: 0..59 Unit: Second Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
LatitudeWholeDegrees Range: 0..89 Unit: Degree Res: 1
MinutesLatLon Range: 0..59.99 Unit: Minute Res: 0.01

Figure 4.6-63. LatitudeType Variable Hierarchy

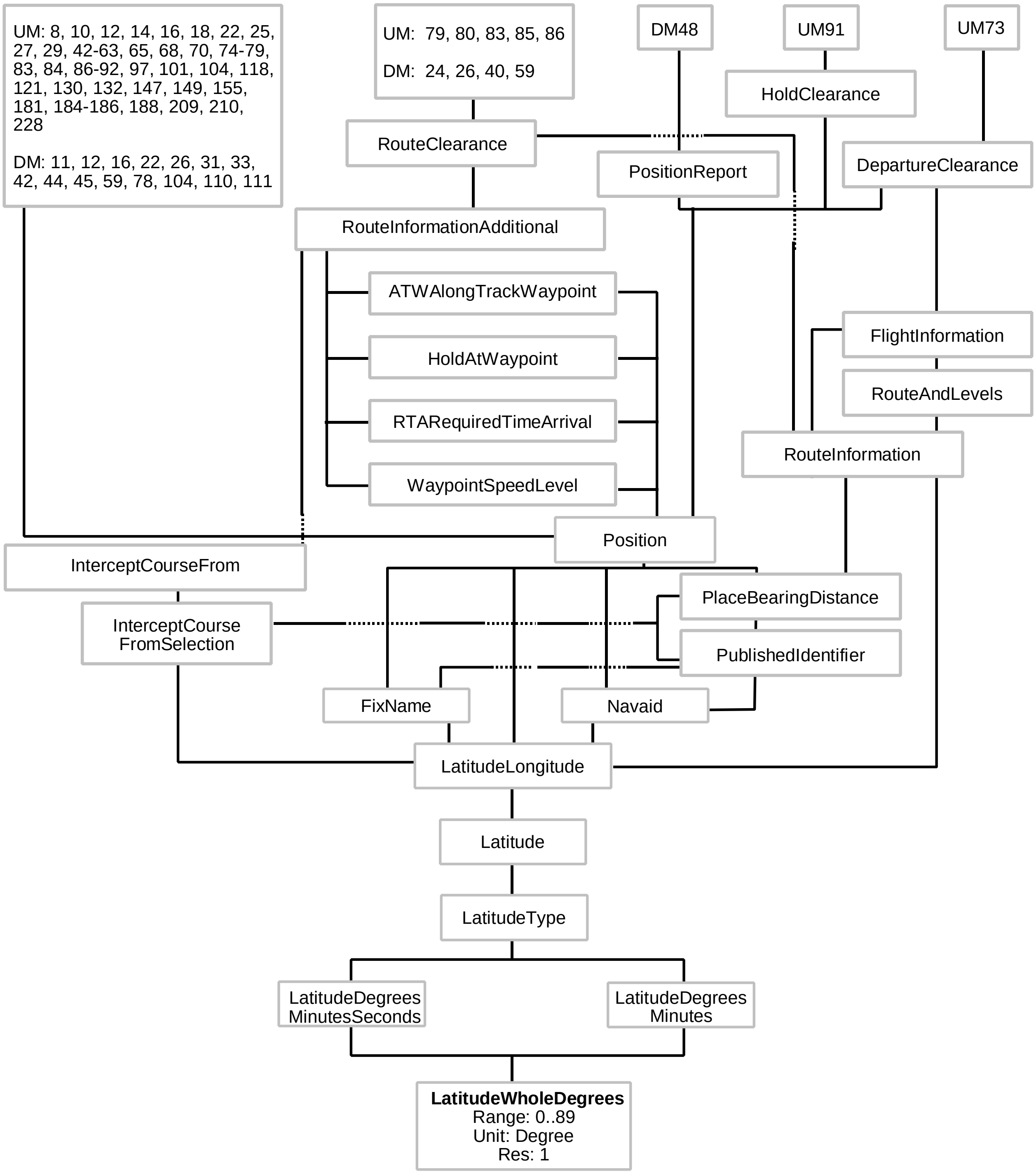


Figure 4.6-64. LatitudeWholeDegrees Variable Hierarchy

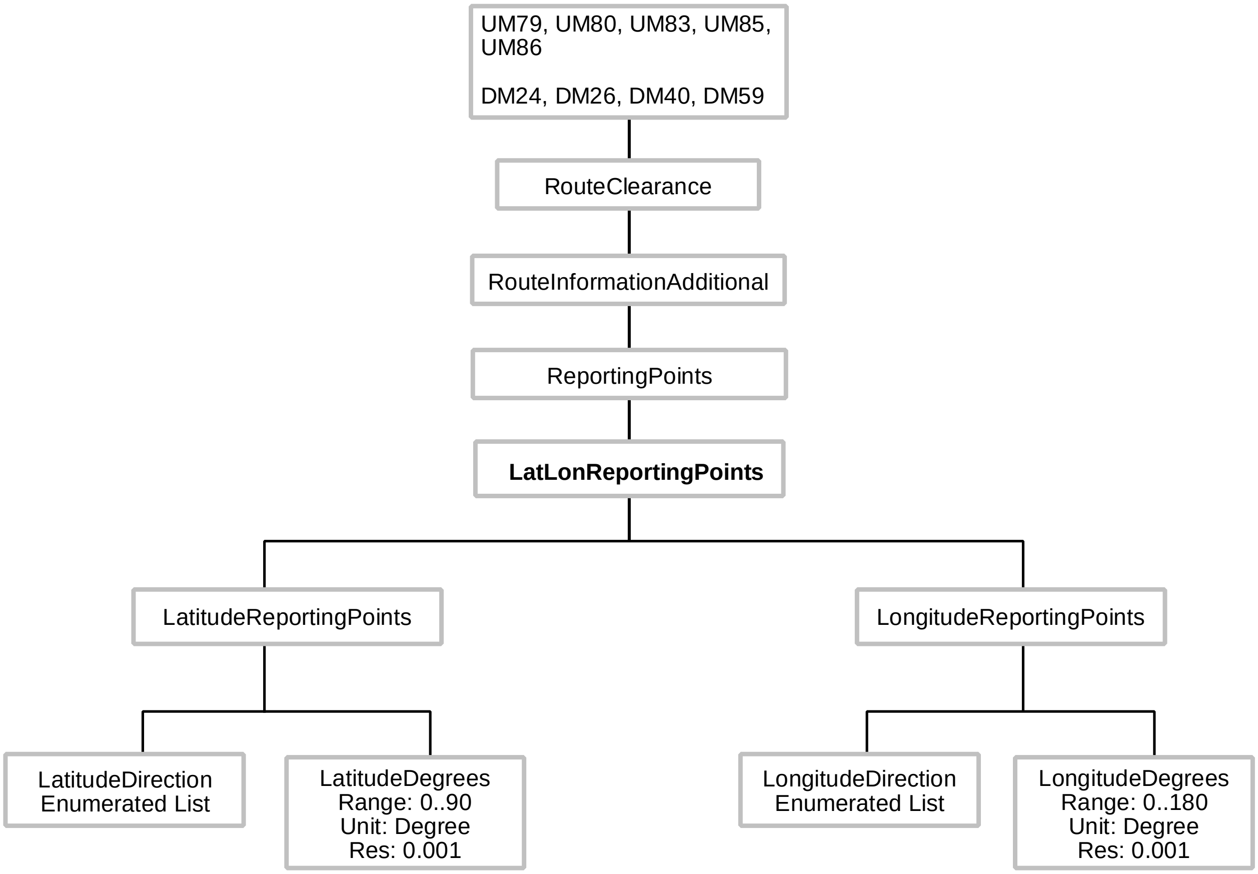

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
ReportingPoints
LatLonReportingPoints
LatitudeReportingPoints
LongitudeReportingPoints
LatitudeDirection Enumerated List
LatitudeDegrees Range: 0..90 Unit: Degree Res: 0.001
LongitudeDirection Enumerated List
LongitudeDegrees Range: 0..180 Unit: Degree Res: 0.001

Figure 4.6-65. LatLonReportingPoints Variable Hierarchy

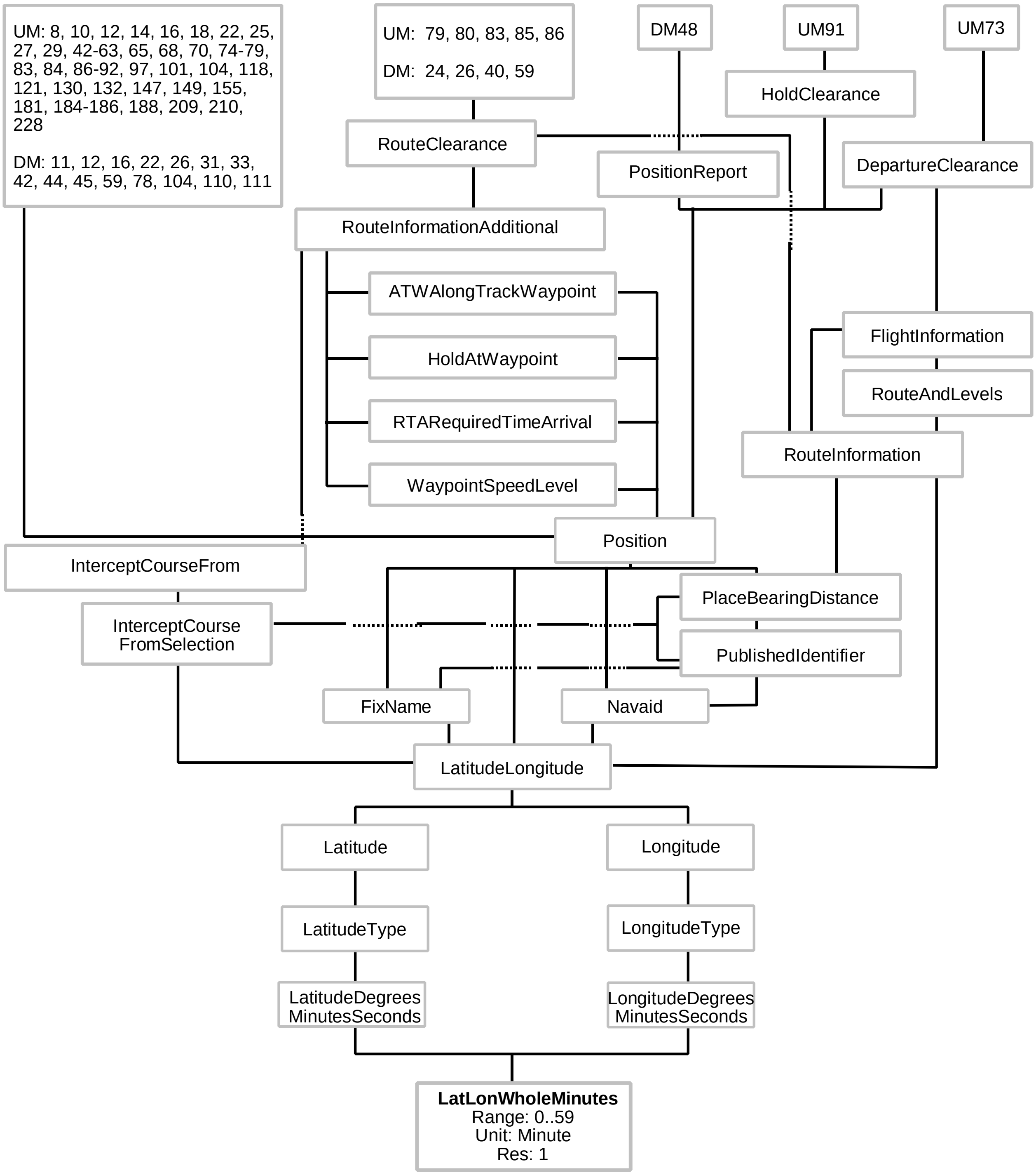


Figure 4.6-66. LatLonWholeMinutes Variable Hierarchy

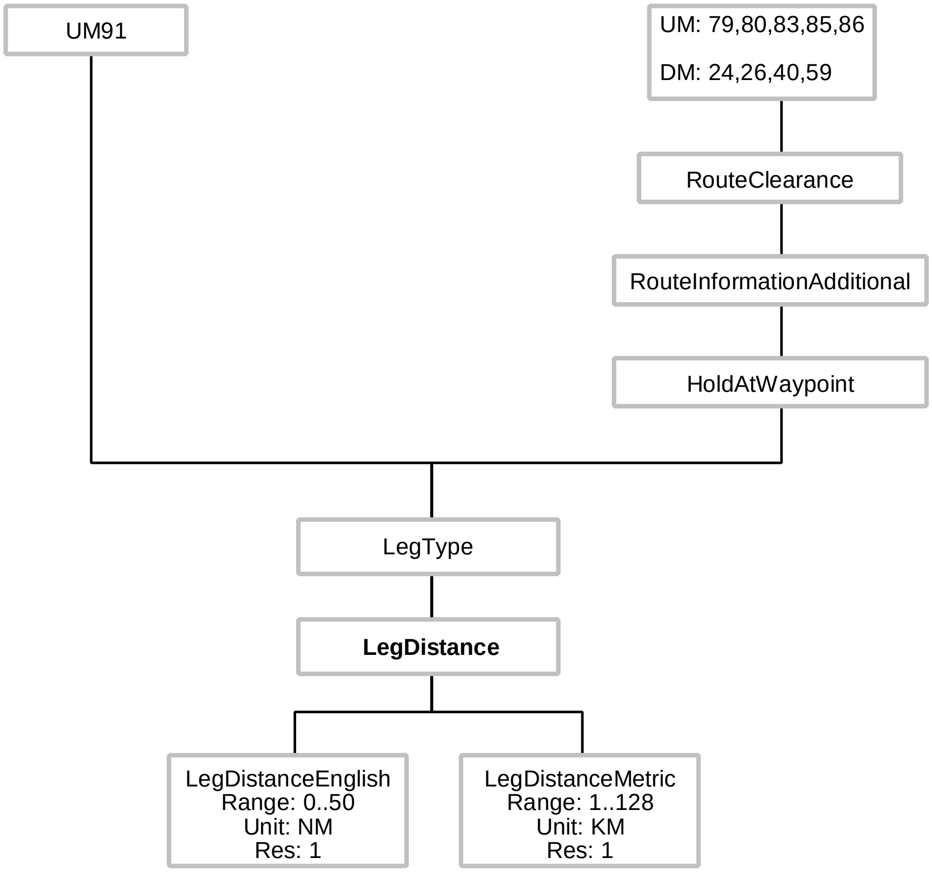


Figure 4.6-67. LegDistance Variable Hierarchy

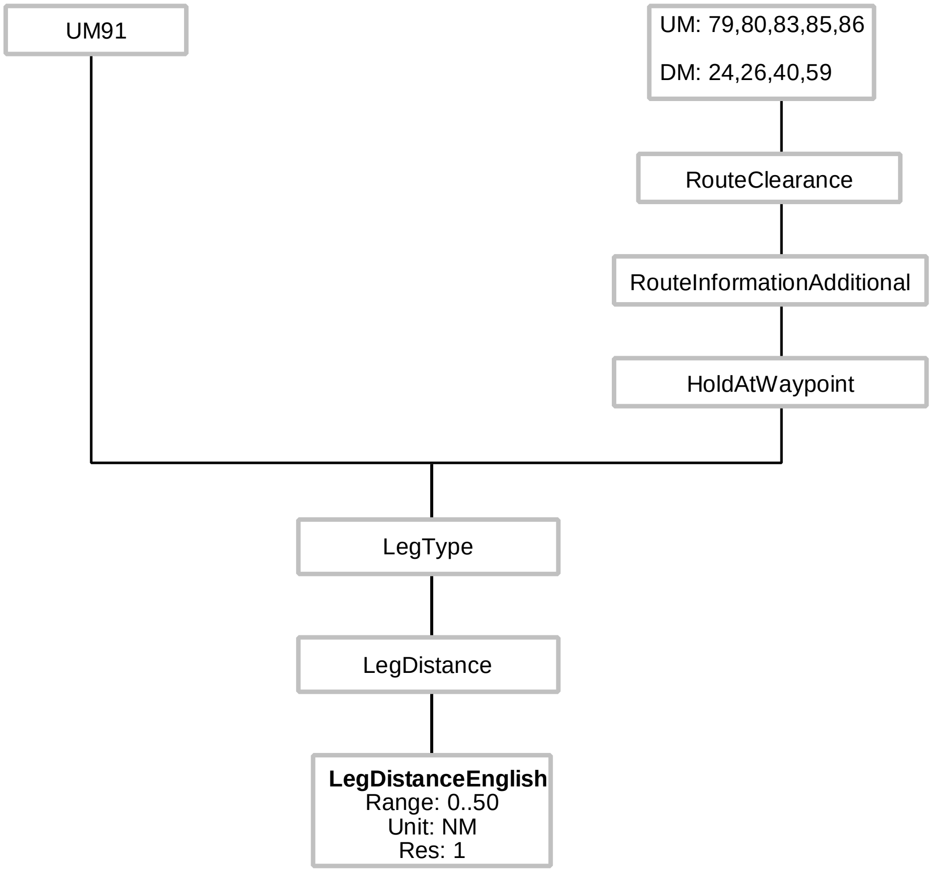


Figure 4.6-68. LegDistanceEnglish Variable Hierarchy

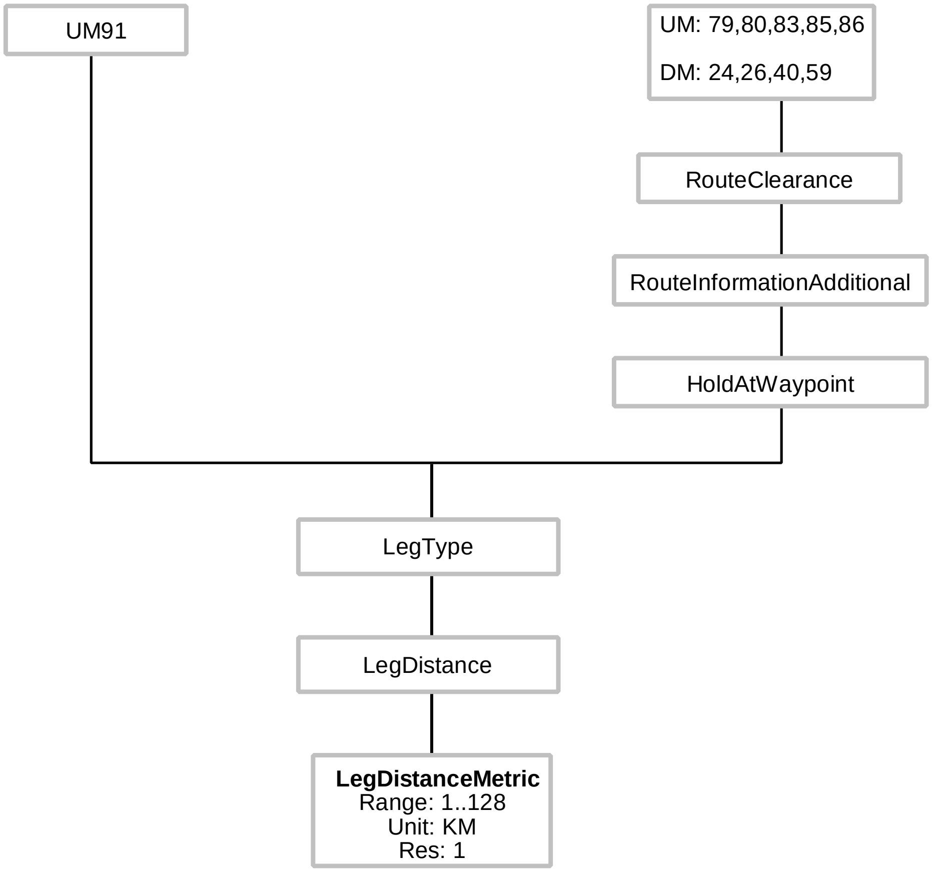


Figure 4.6-69. LegDistanceMetric Variable Hierarchy

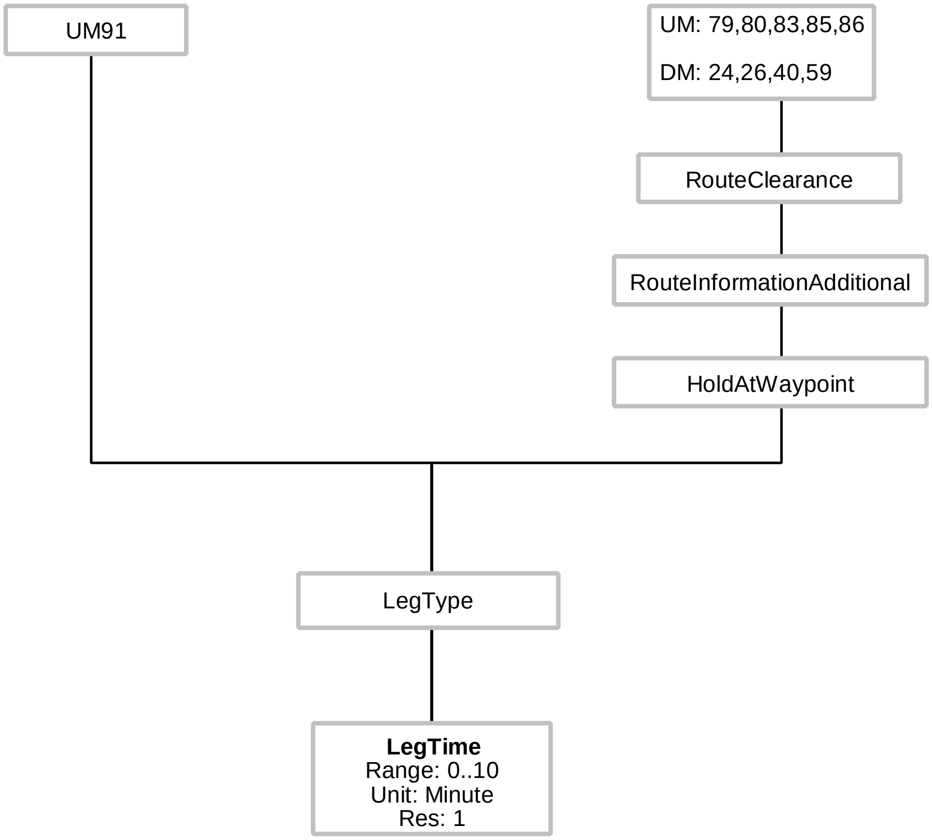


Figure 4.6-70. LegTime Variable Hierarchy

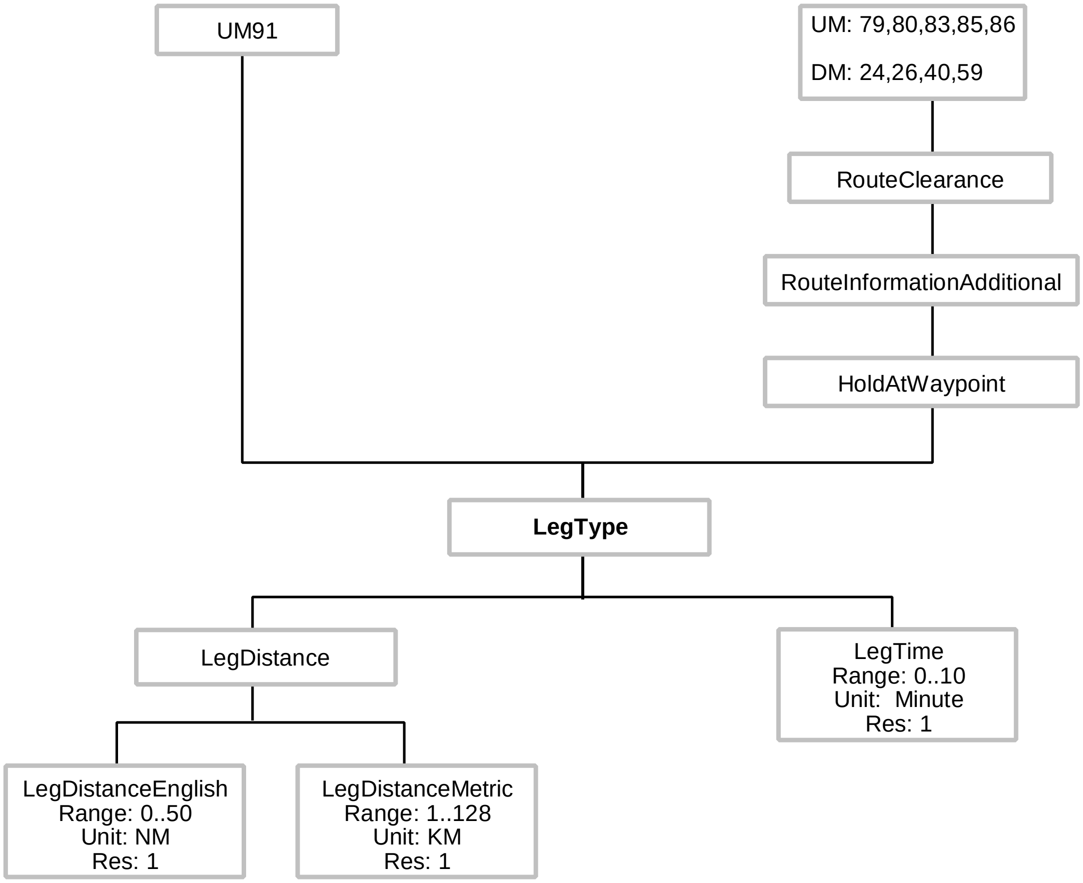


Figure 4.6-71. LegType Variable Hierarchy

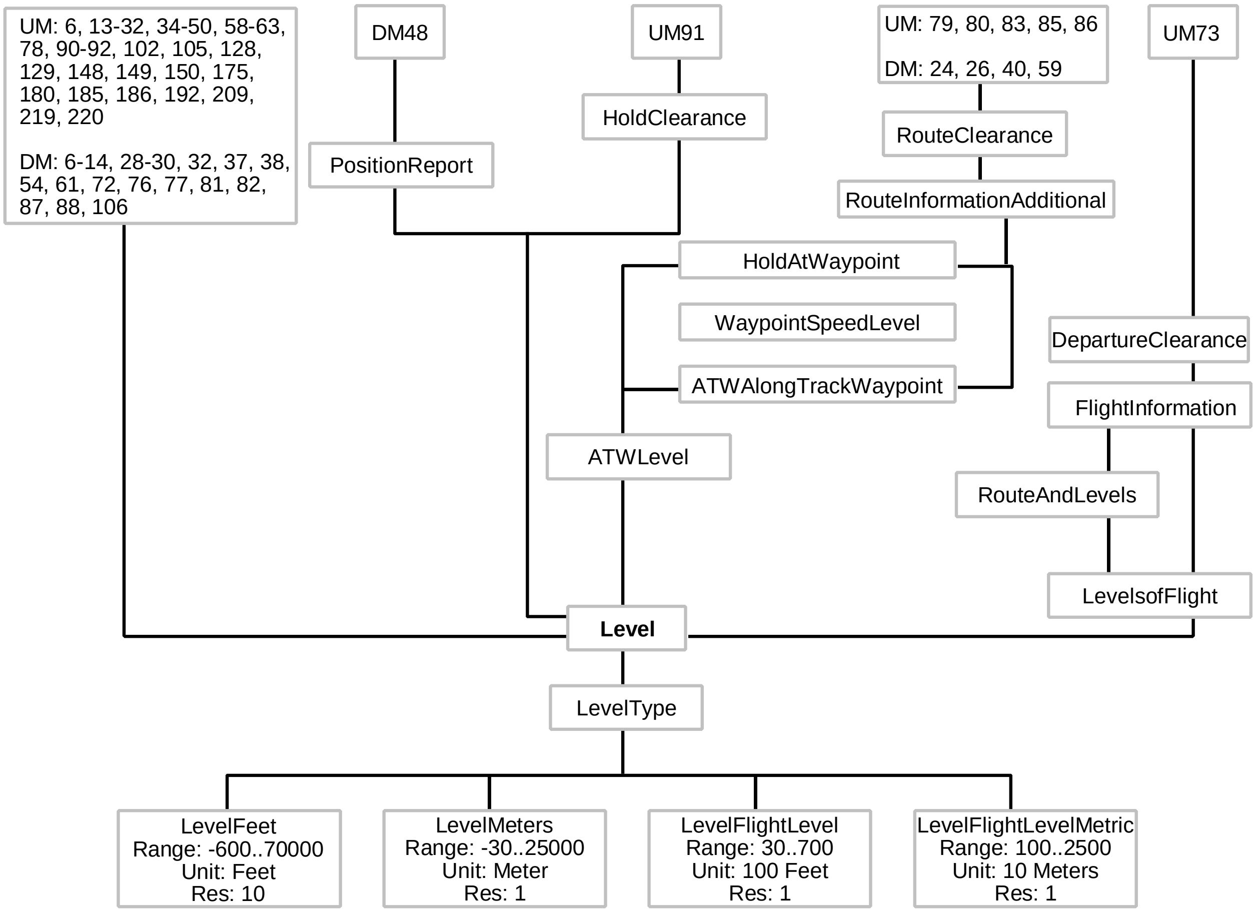


Figure 4.6-72. Level Variable Hierarchy

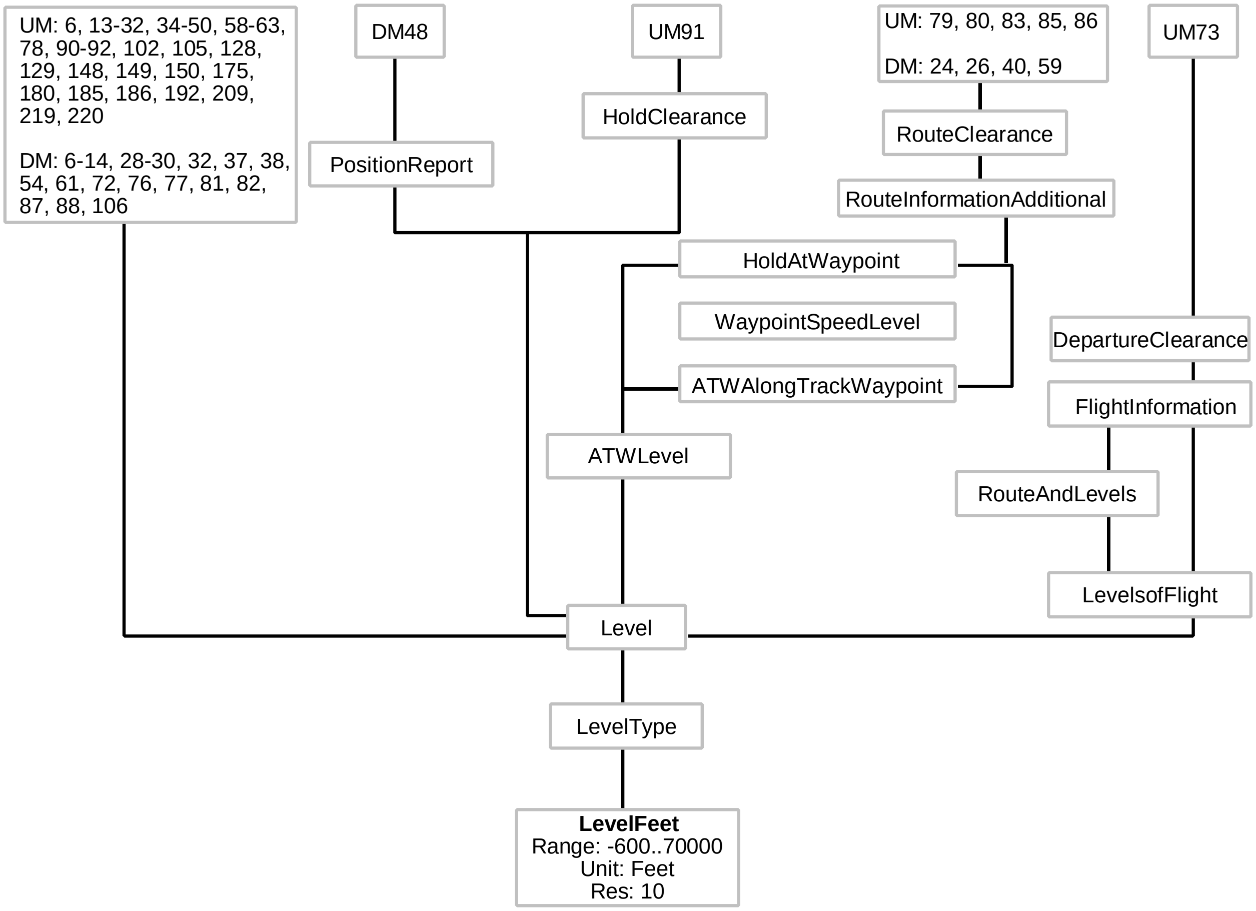


Figure 4.6-73. LevelFeet Variable Hierarchy

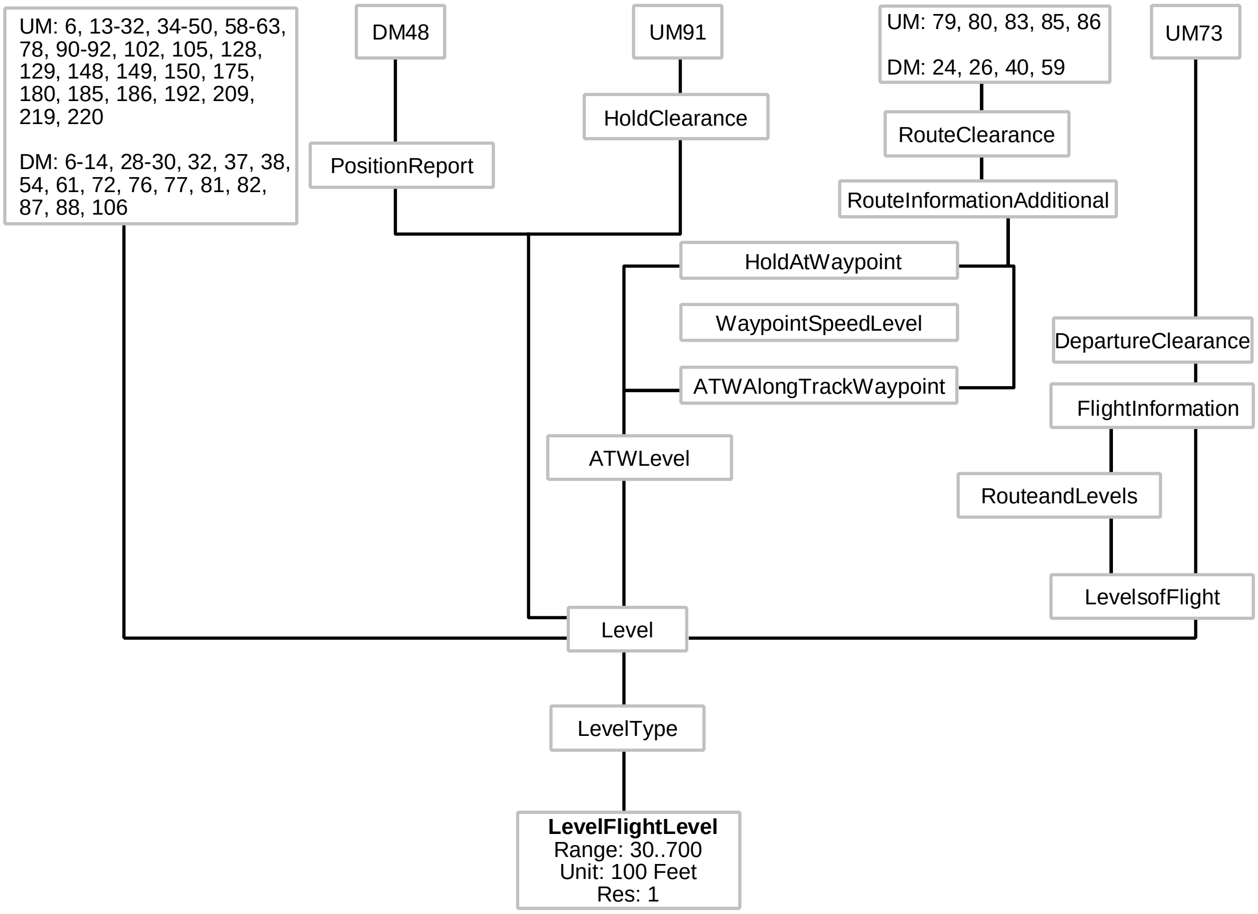


Figure 4.6-74. LevelFlightLevel Variable Hierarchy

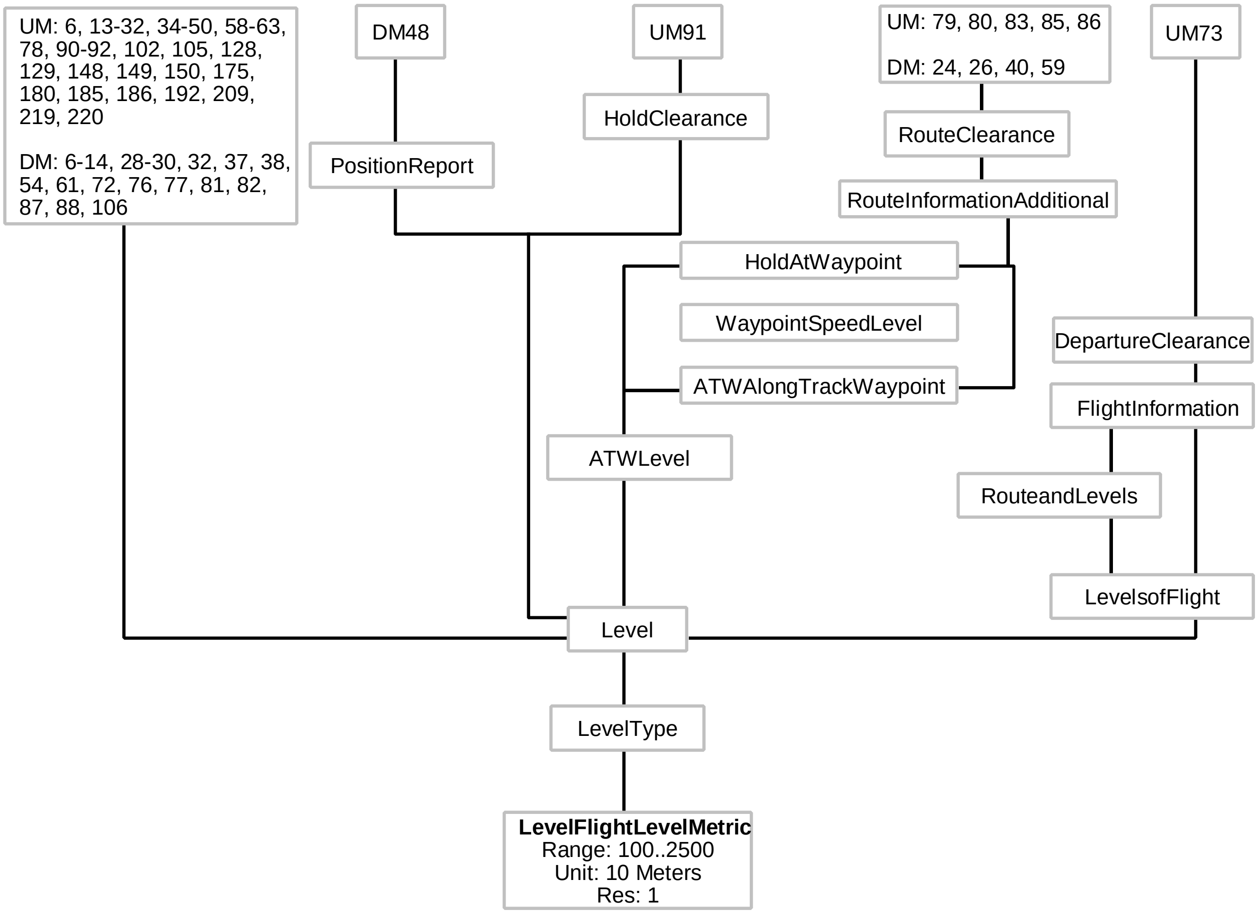


Figure 4.6-75. LevelFlightLevelMetric Variable Hierarchy

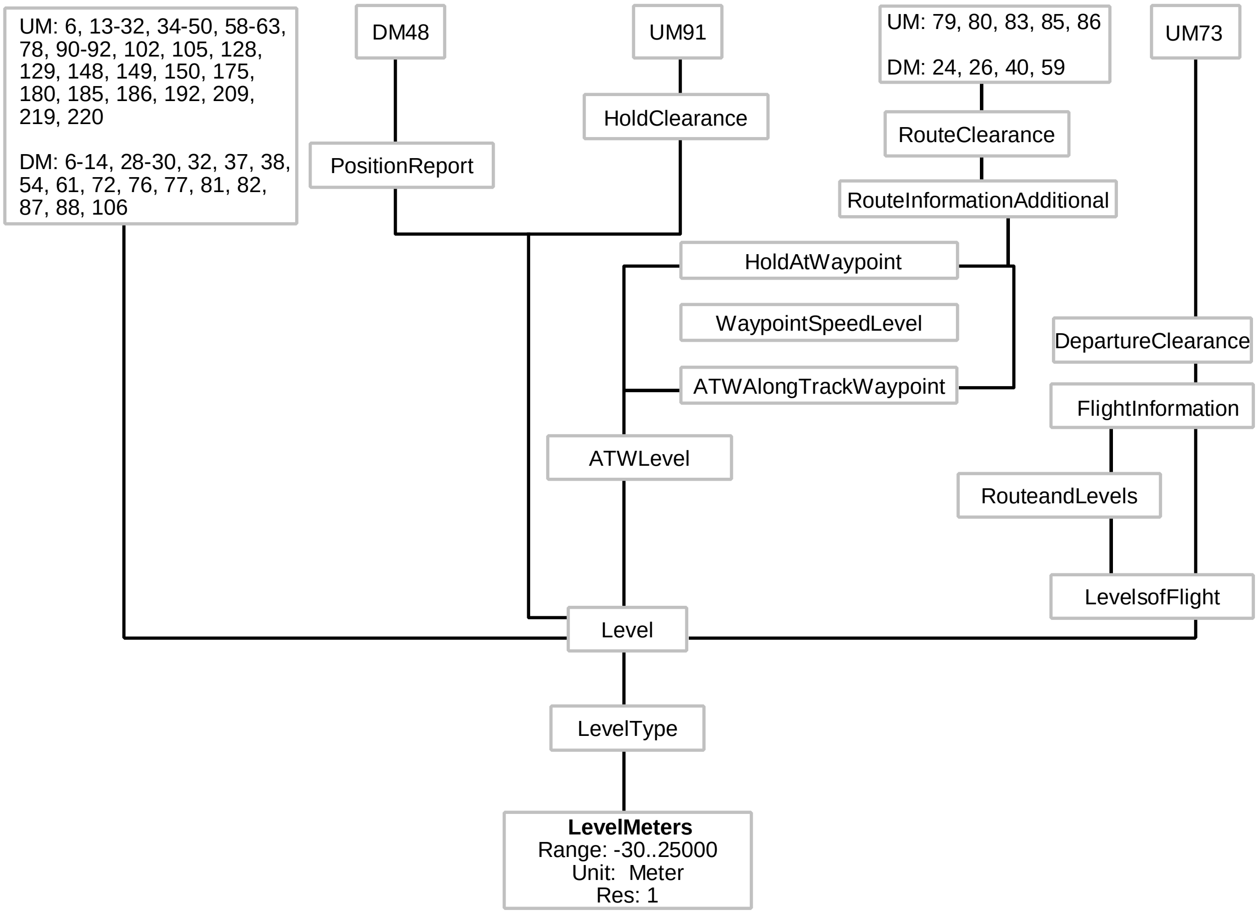


Figure 4.6-76. LevelMetric Variable Hierarchy

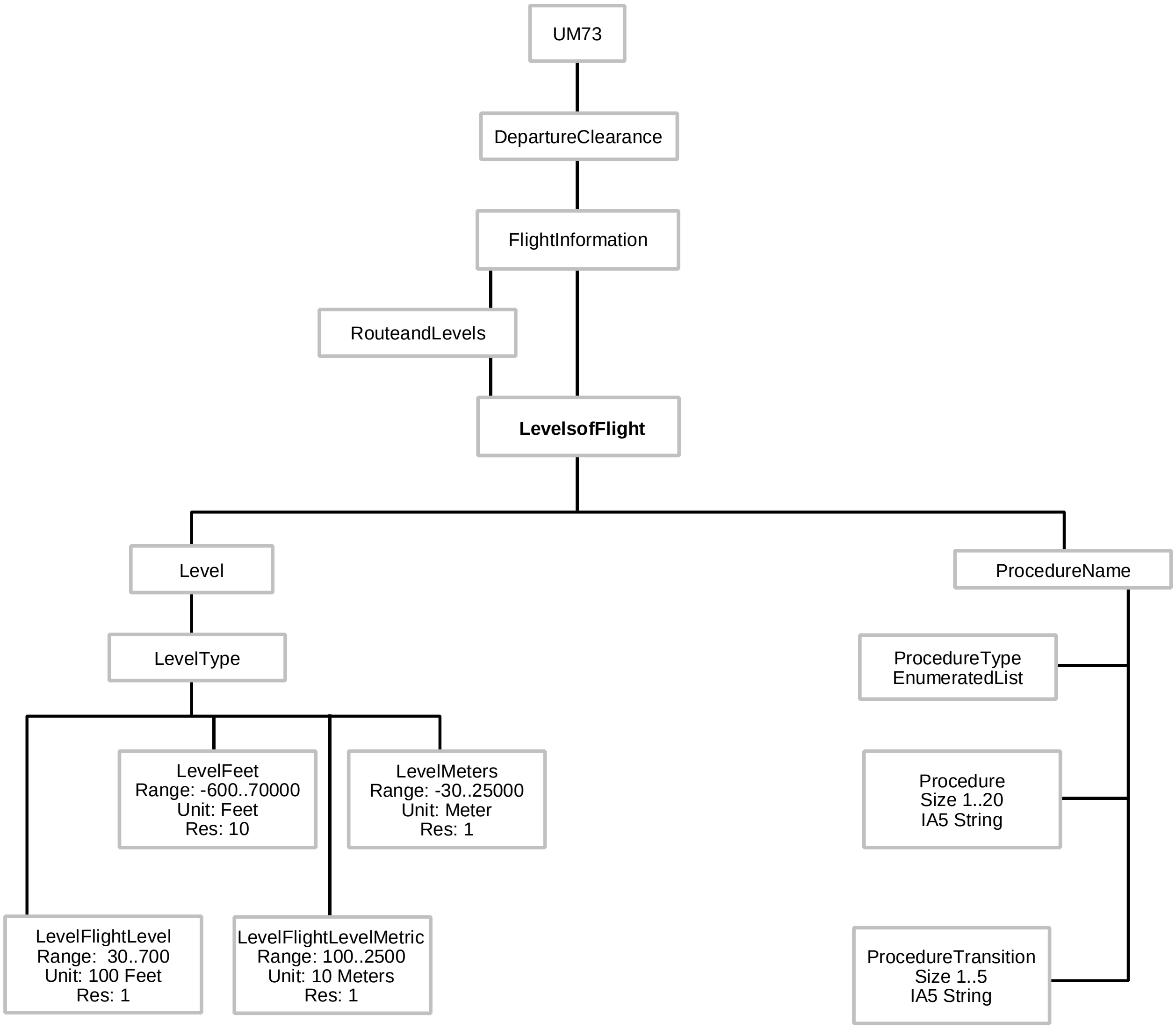

UM73
DepartureClearance
FlightInformation
RouteandLevels
LevelsofFlight
Level
LevelType
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
ProcedureName
ProcedureType EnumeratedList
Procedure Size 1..20 IA5 String
ProcedureTransition Size 1..5 IA5 String

Figure 4.6-77. LevelsofFlight Variable Hierarchy

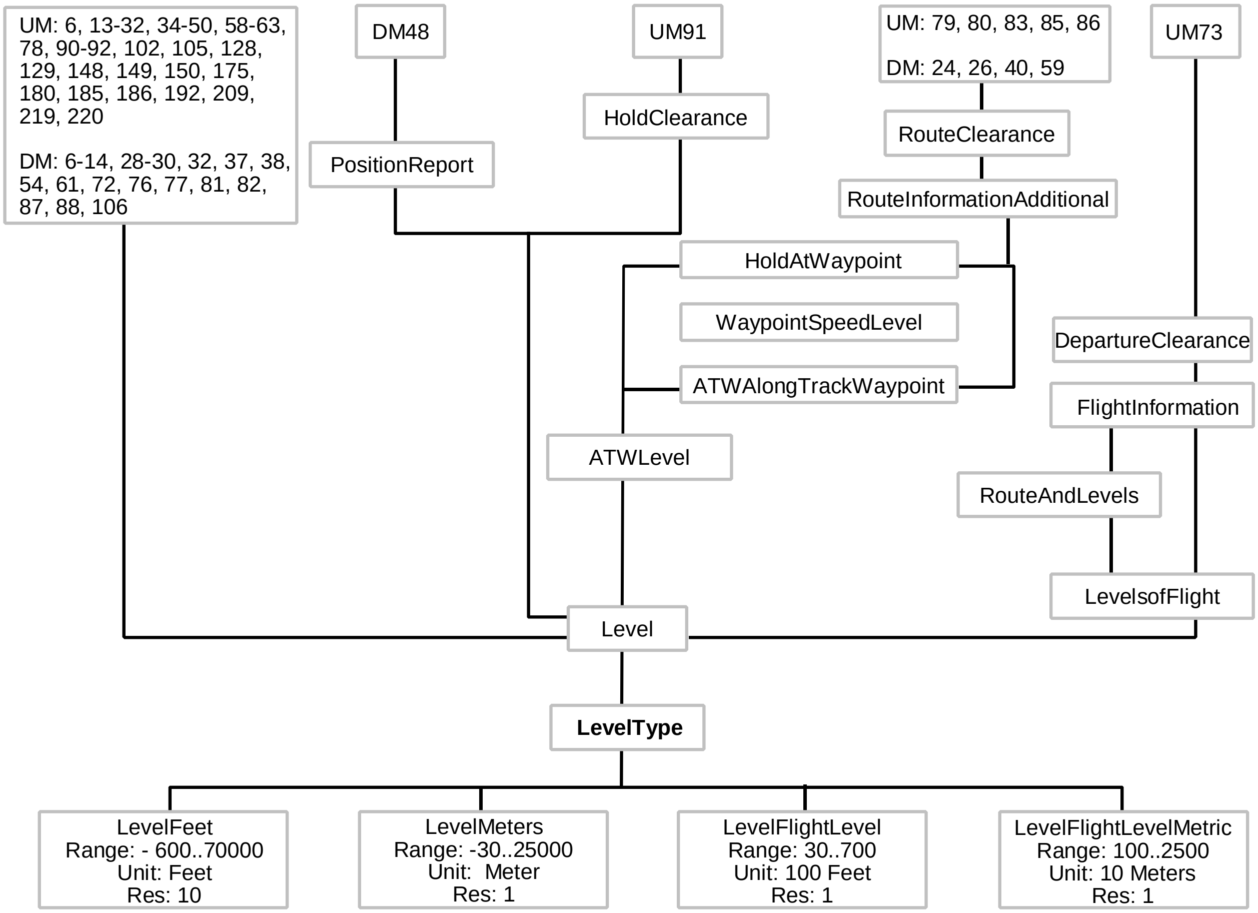


Figure 4.6-78. LevelType Variable Hierarchy

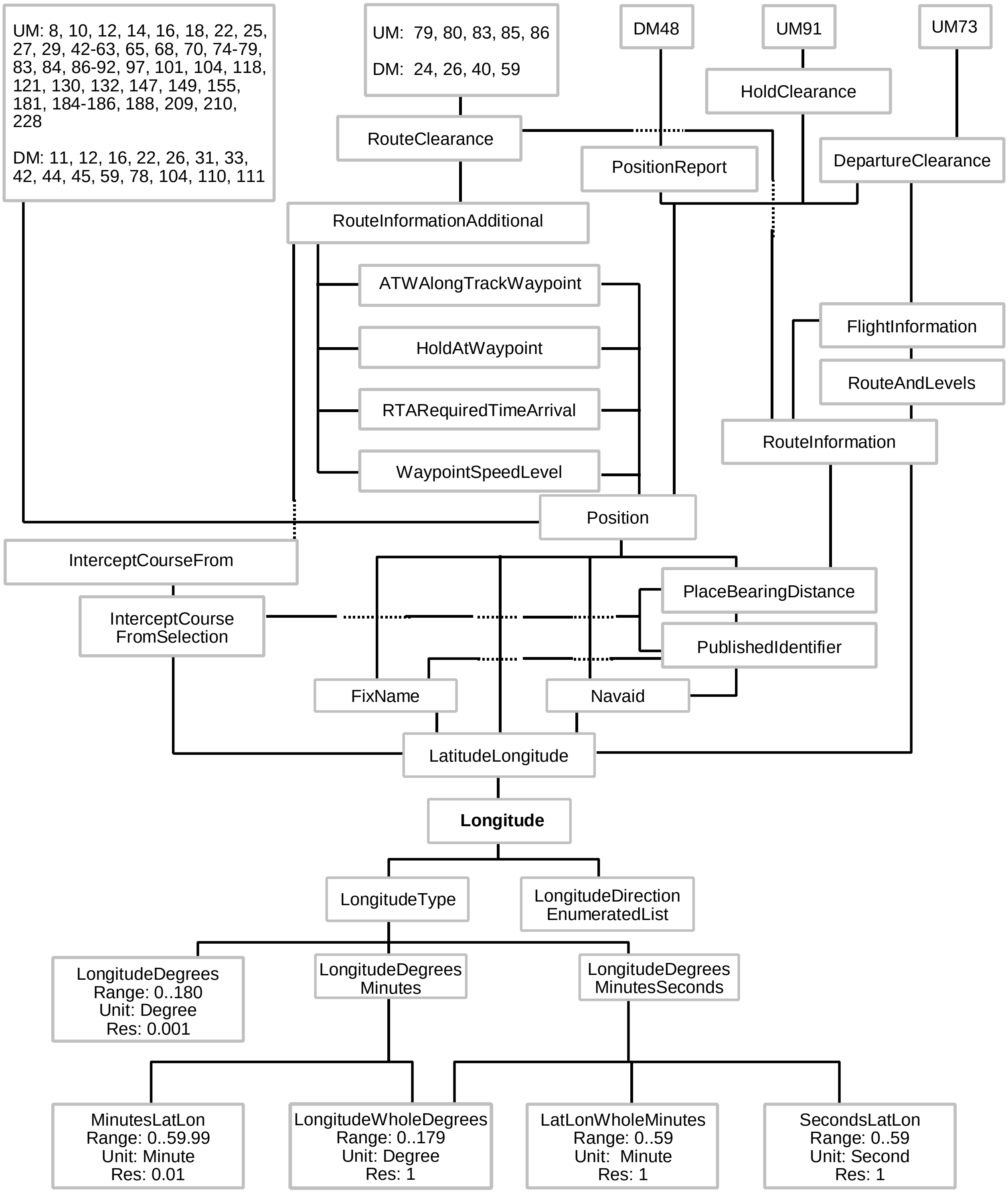

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 74-79, 83, 84, 86-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 26, 31, 33, 42, 44, 45, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
UM91
UM73
HoldClearance
RouteClearance
PositionReport
DepartureClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldAtWaypoint
RTARequiredTimeArrival
WaypointSpeedLevel
FlightInformation
RouteAndLevels
RouteInformation
Position
InterceptCourseFrom
InterceptCourse FromSelection
PlaceBearingDistance
PublishedIdentifier
FixName
Navaid
LatitudeLongitude
Longitude
LongitudeType
LongitudeDirection EnumeratedList
LongitudeDegrees Range: 0..180 Unit: Degree Res: 0.001
LongitudeDegrees Minutes
LongitudeDegrees MinutesSeconds
MinutesLatLon Range: 0..59.99 Unit: Minute Res: 0.01
LongitudeWholeDegrees Range: 0..179 Unit: Degree Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
SecondsLatLon Range: 0..59 Unit: Second Res: 1

Figure 4.6-79. Longitude Variable Hierarchy

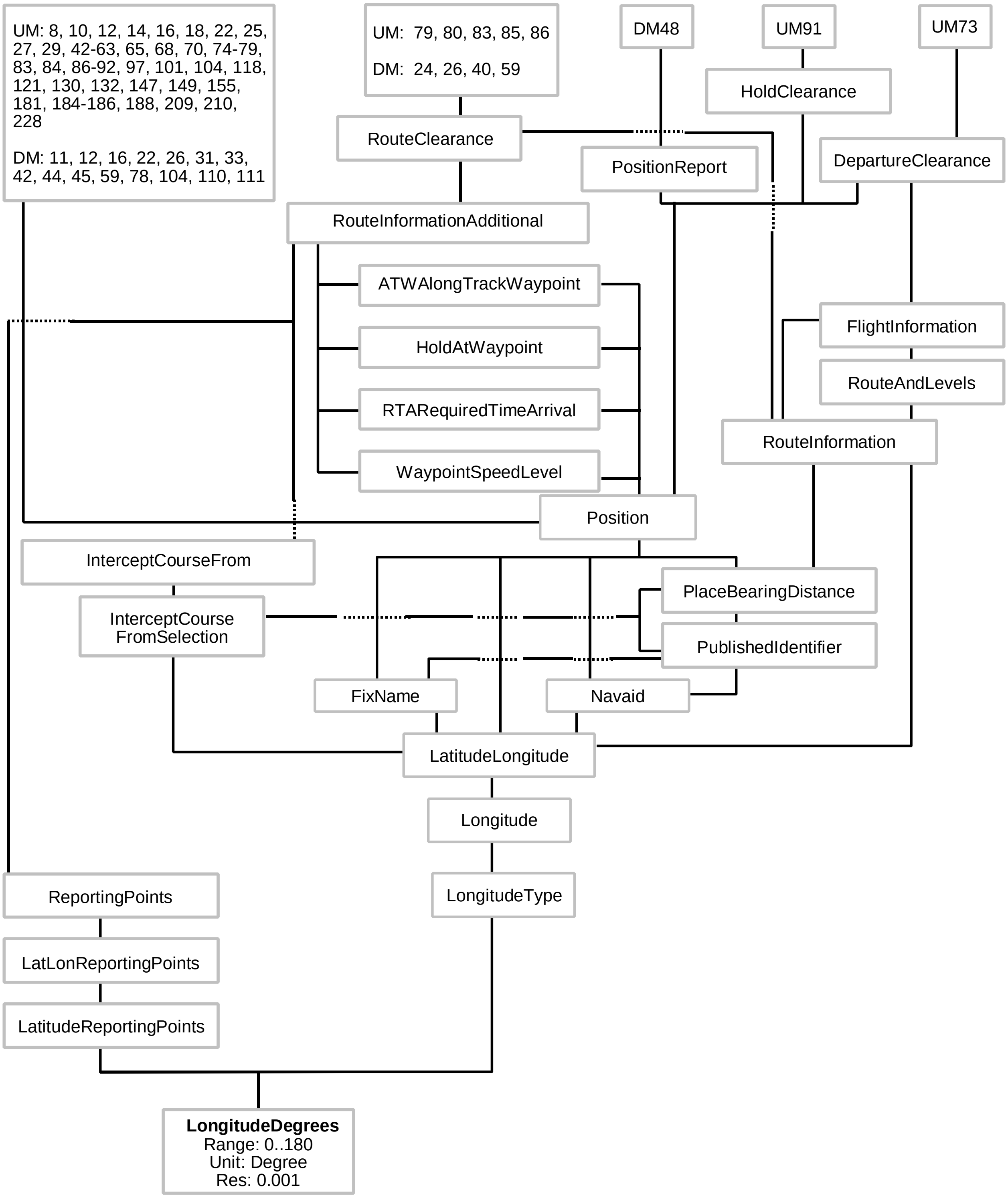


Figure 4.6-80. LongitudeDegrees Variable Hierarchy

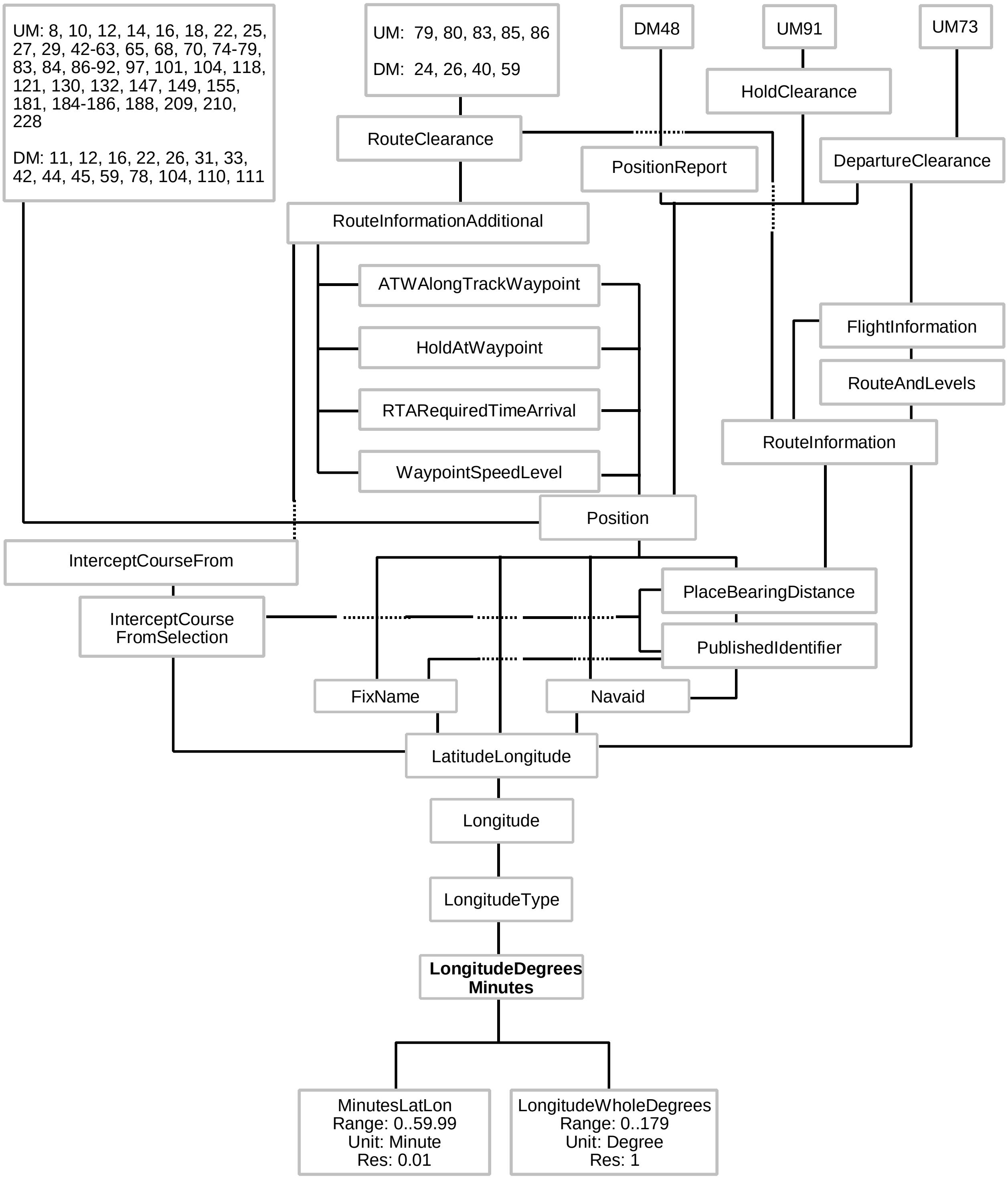


Figure 4.6-81. LongitudeDegreesMinutes Variable Hierarchy

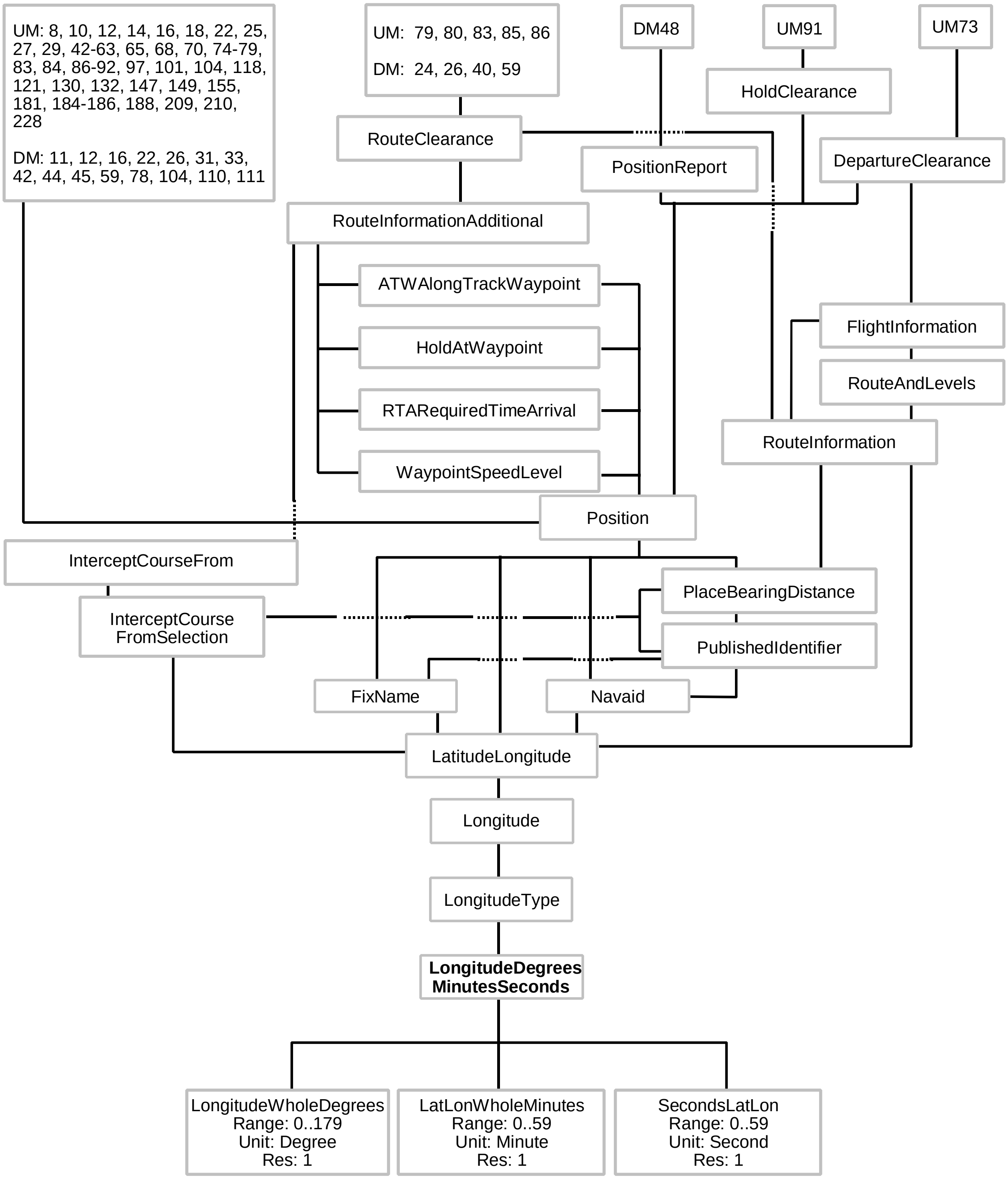

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 74-79, 83, 84, 86-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 26, 31, 33, 42, 44, 45, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
UM91
UM73
HoldClearance
RouteClearance
PositionReport
DepartureClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldAtWaypoint
RTARequiredTimeArrival
WaypointSpeedLevel
FlightInformation
RouteAndLevels
RouteInformation
Position
InterceptCourseFrom
InterceptCourse FromSelection
PlaceBearingDistance
PublishedIdentifier
FixName
Navaid
LatitudeLongitude
Longitude
LongitudeType
LongitudeDegrees MinutesSeconds
LongitudeWholeDegrees Range: 0..179 Unit: Degree Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
SecondsLatLon Range: 0..59 Unit: Second Res: 1

Figure 4.6-82. LongitudeDegreesMinutesSeconds Variable Hierarchy

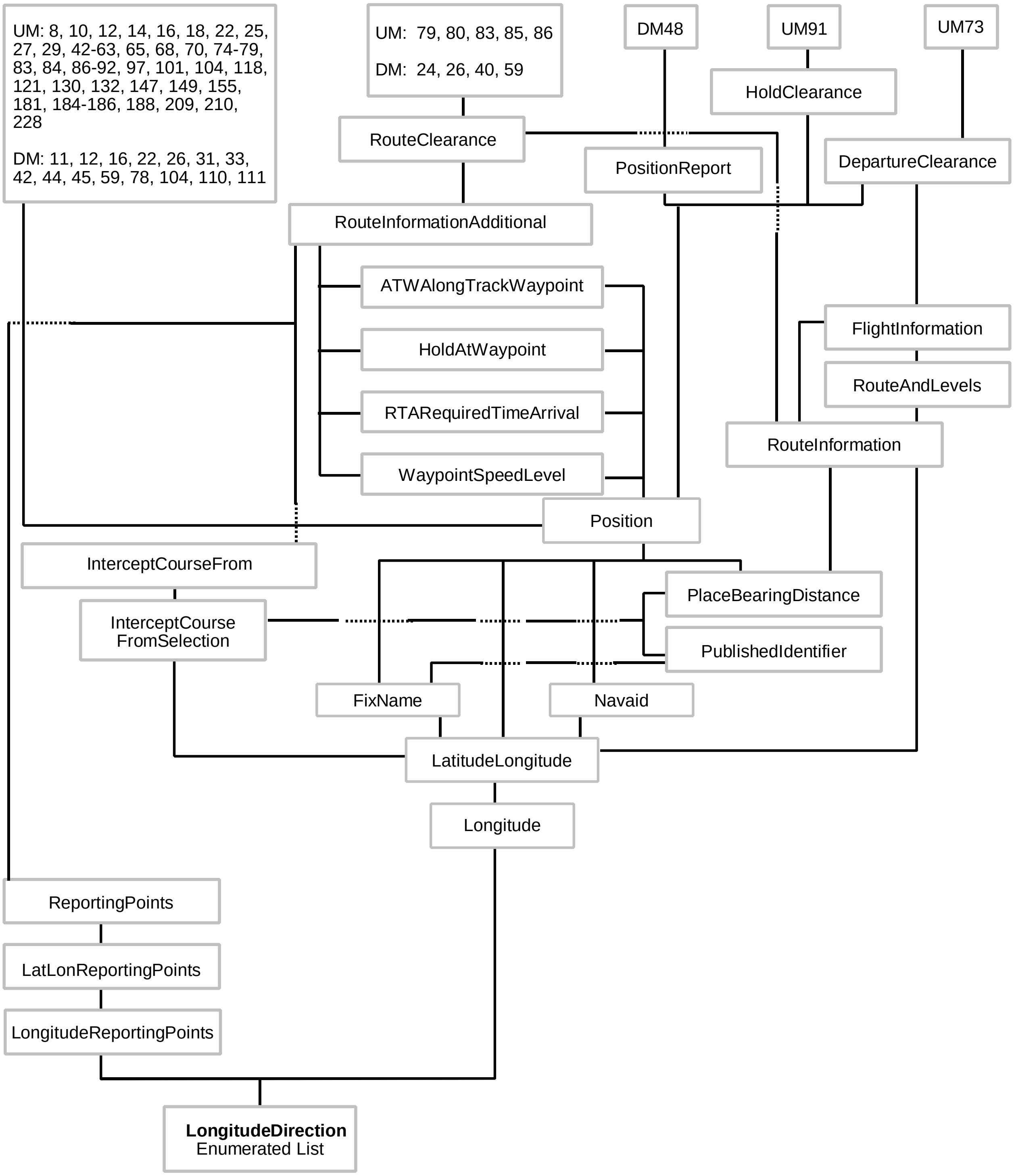


Figure 4.6-83. LongitudeDirection Variable Hierarchy

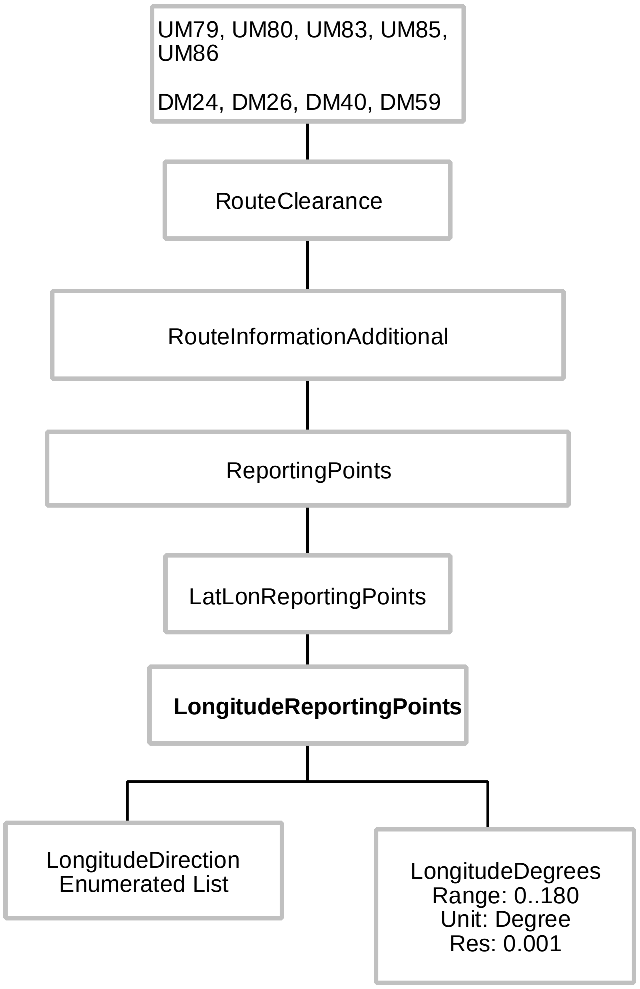

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
ReportingPoints
LatLonReportingPoints
LongitudeReportingPoints
LongitudeDirection Enumerated List
LongitudeDegrees Range: 0..180 Unit: Degree Res: 0.001

Figure 4.6-84. LongitudeReportingPoints Variable Hierarchy

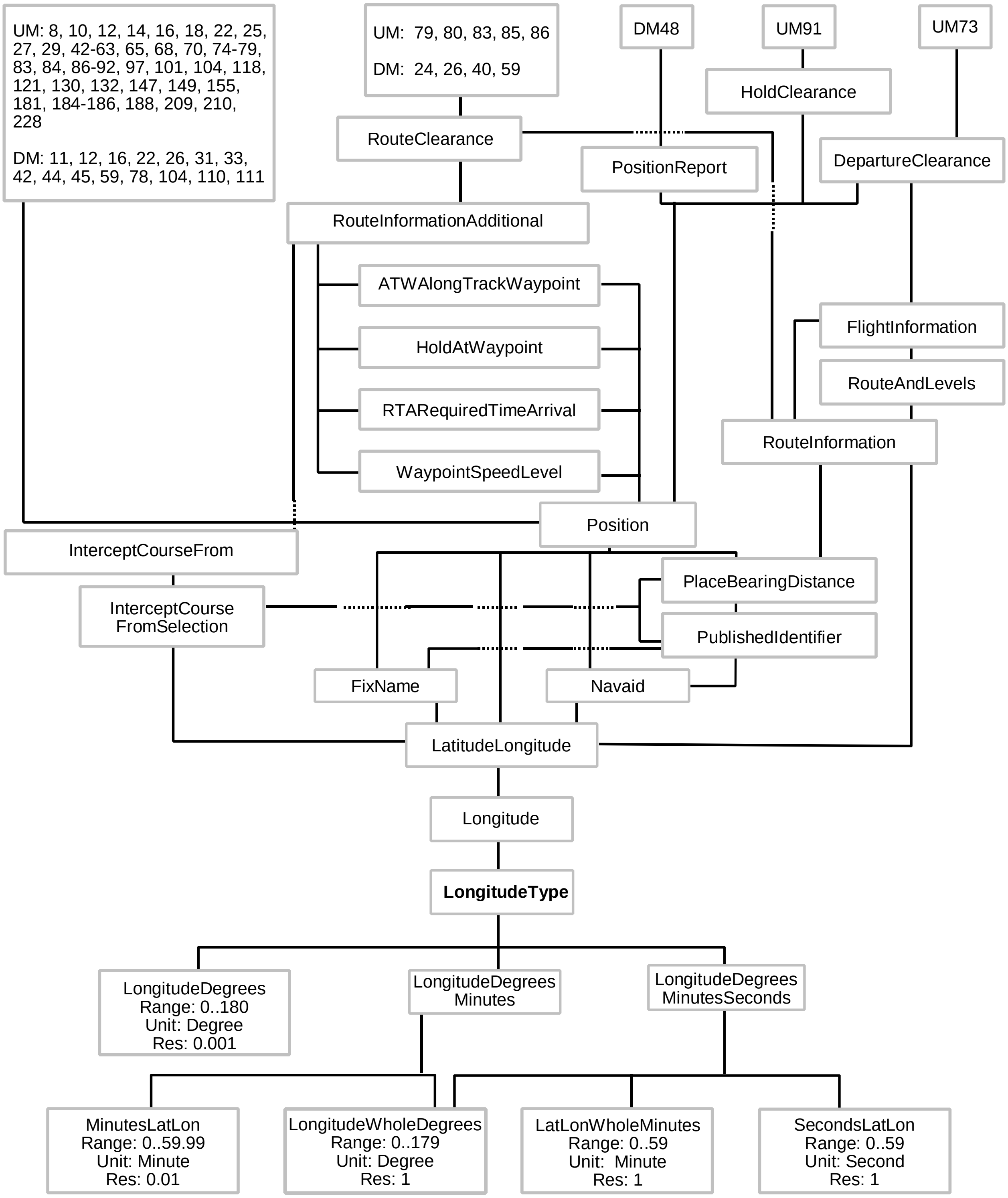

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 74-79, 83, 84, 86-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 26, 31, 33, 42, 44, 45, 59, 78, 104, 110, 111
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
UM91
UM73
HoldClearance
RouteClearance
PositionReport
DepartureClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
HoldAtWaypoint
RTARequiredTimeArrival
WaypointSpeedLevel
FlightInformation
RouteAndLevels
RouteInformation
Position
InterceptCourseFrom
InterceptCourse FromSelection
PlaceBearingDistance
PublishedIdentifier
FixName
Navaid
LatitudeLongitude
Longitude
LongitudeType
LongitudeDegrees Range: 0..180 Unit: Degree Res: 0.001
LongitudeDegrees Minutes
LongitudeDegrees MinutesSeconds
MinutesLatLon Range: 0..59.99 Unit: Minute Res: 0.01
LongitudeWholeDegrees Range: 0..179 Unit: Degree Res: 1
LatLonWholeMinutes Range: 0..59 Unit: Minute Res: 1
SecondsLatLon Range: 0..59 Unit: Second Res: 1

Figure 4.6-85. LongitudeType Variable Hierarchy

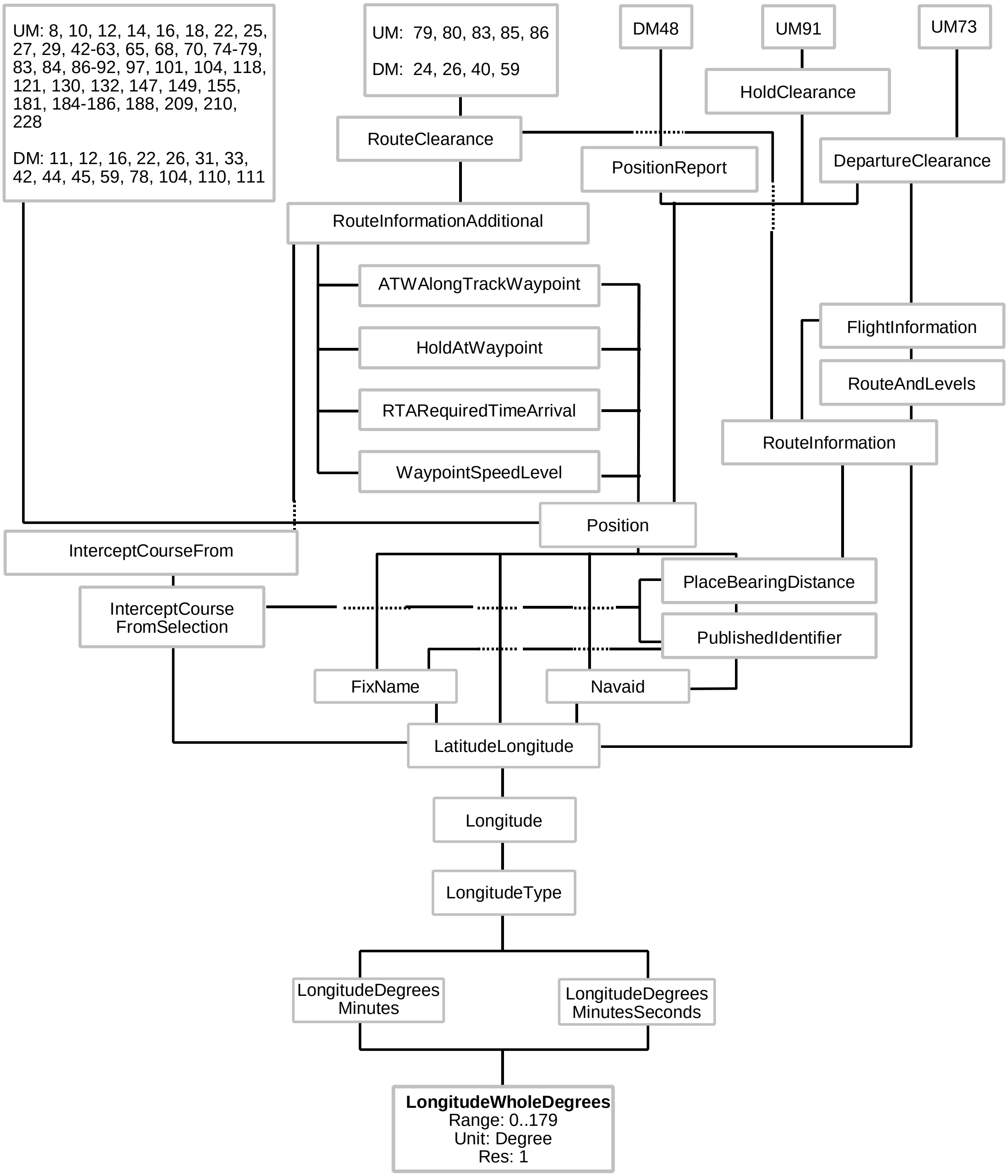


Figure 4.6-86. LongitudeWholeDegrees Variable Hierarchy

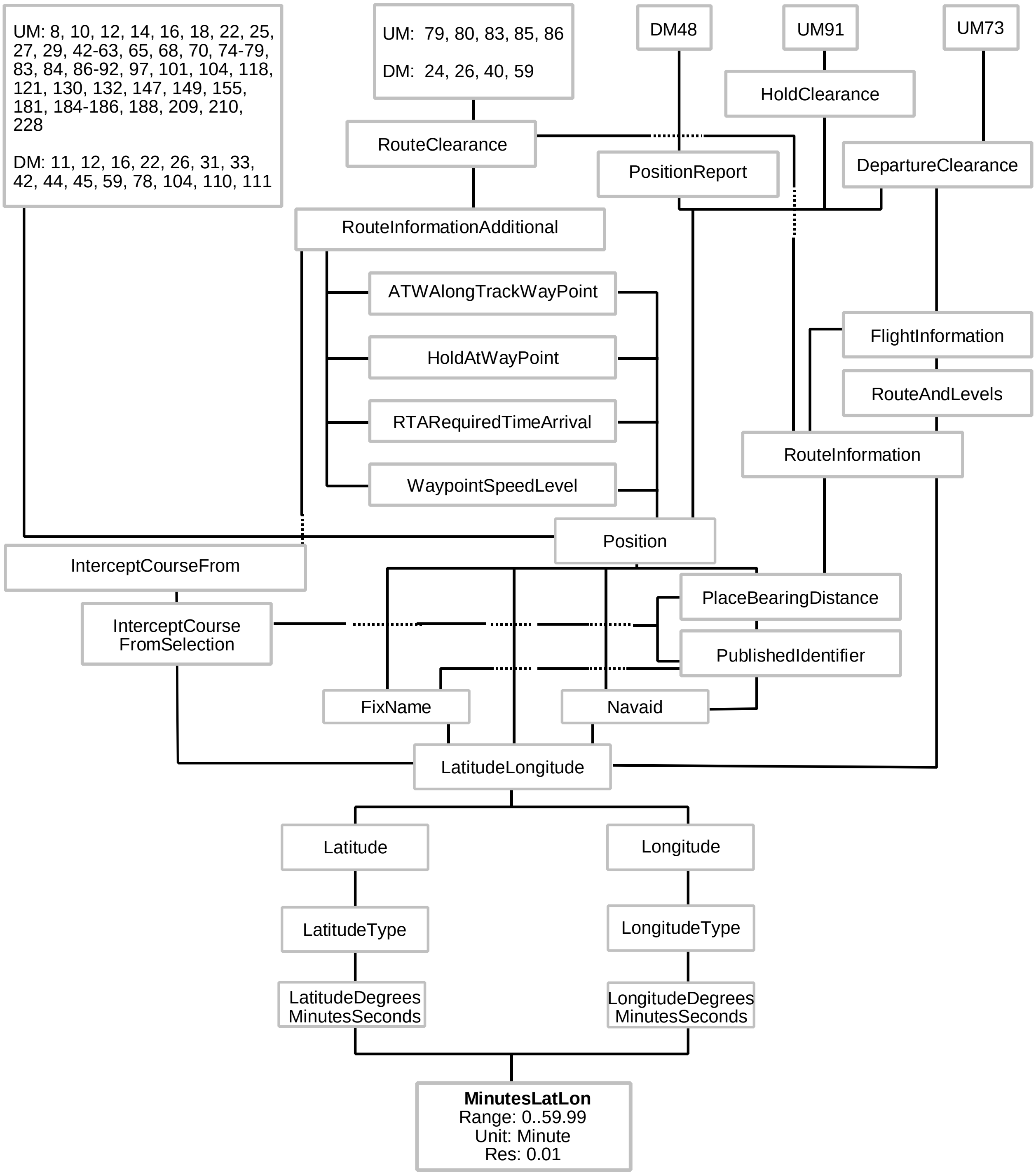


Figure 4.6-87. MinutesLatLon Variable Hierarchy

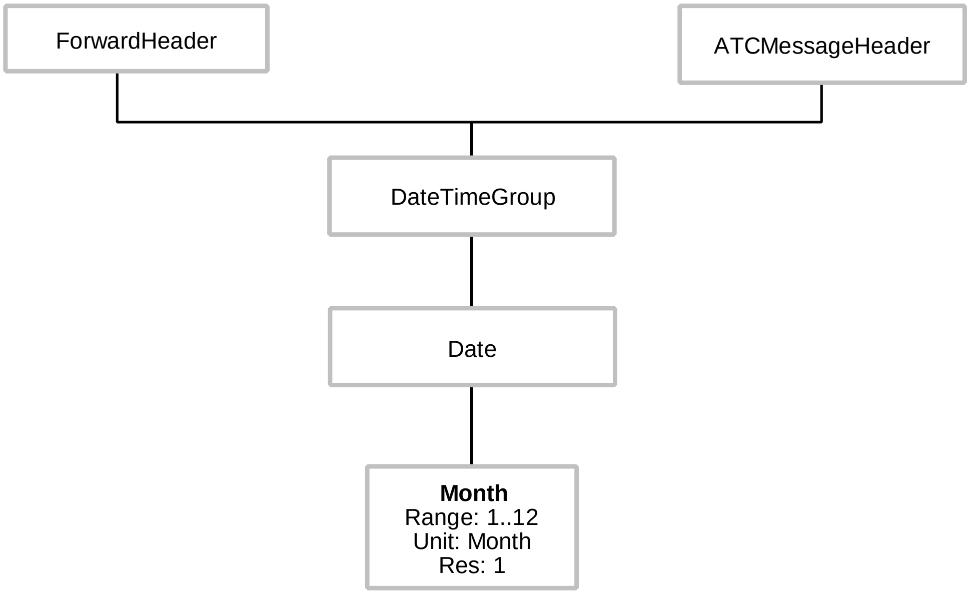


Figure 4.6-88. Month Variable Hierarchy

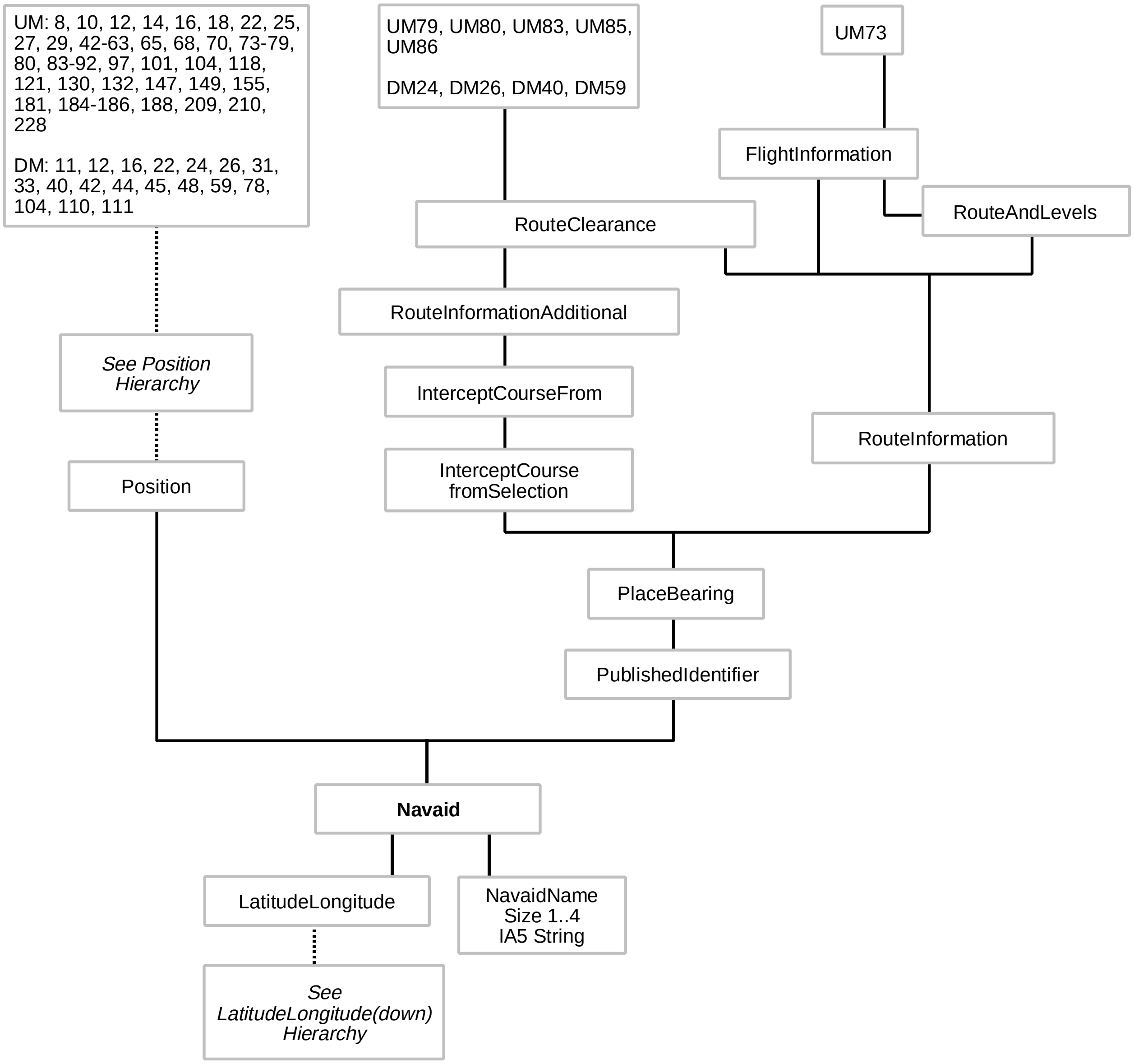

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 73-79, 80, 83-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 24, 26, 31, 33, 40, 42, 44, 45, 48, 59, 78, 104, 110, 111
UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM73
FlightInformation
RouteClearance
RouteAndLevels
RouteInformationAdditional
See Position Hierarchy
InterceptCourseFrom
RouteInformation
Position
InterceptCourse fromSelection
PlaceBearing
PublishedIdentifier
Navaid
LatitudeLongitude
NavaidName Size 1..4 IA5 String
See LatitudeLongitude(down) Hierarchy

Figure 4.6-89. Navaid Variable Hierarchy

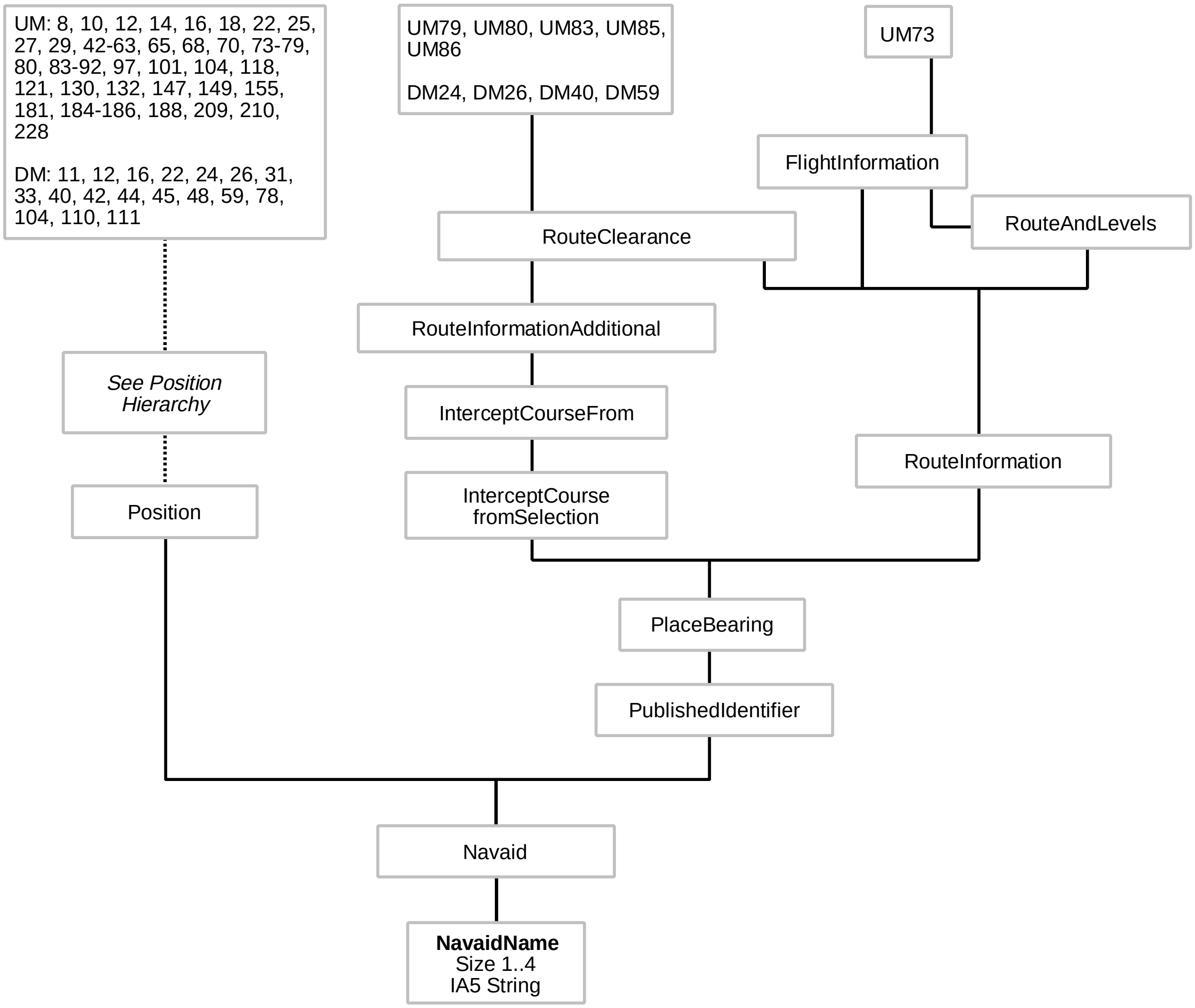

UM: 8, 10, 12, 14, 16, 18, 22, 25, 27, 29, 42-63, 65, 68, 70, 73-79, 80, 83-92, 97, 101, 104, 118, 121, 130, 132, 147, 149, 155, 181, 184-186, 188, 209, 210, 228
DM: 11, 12, 16, 22, 24, 26, 31, 33, 40, 42, 44, 45, 48, 59, 78, 104, 110, 111
UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM73
FlightInformation
RouteAndLevels
RouteClearance
RouteInformationAdditional
See Position Hierarchy
InterceptCourseFrom
RouteInformation
Position
InterceptCourse fromSelection
PlaceBearing
PublishedIdentifier
Navaid
NavaidName
Size 1..4
IA5 String

Figure 4.6-90. NavaidName Variable Hierarchy

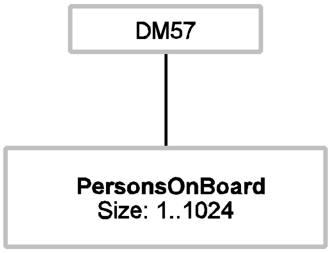


Figure 4.6-91. PersonsOnBoard Variable Hierarchy

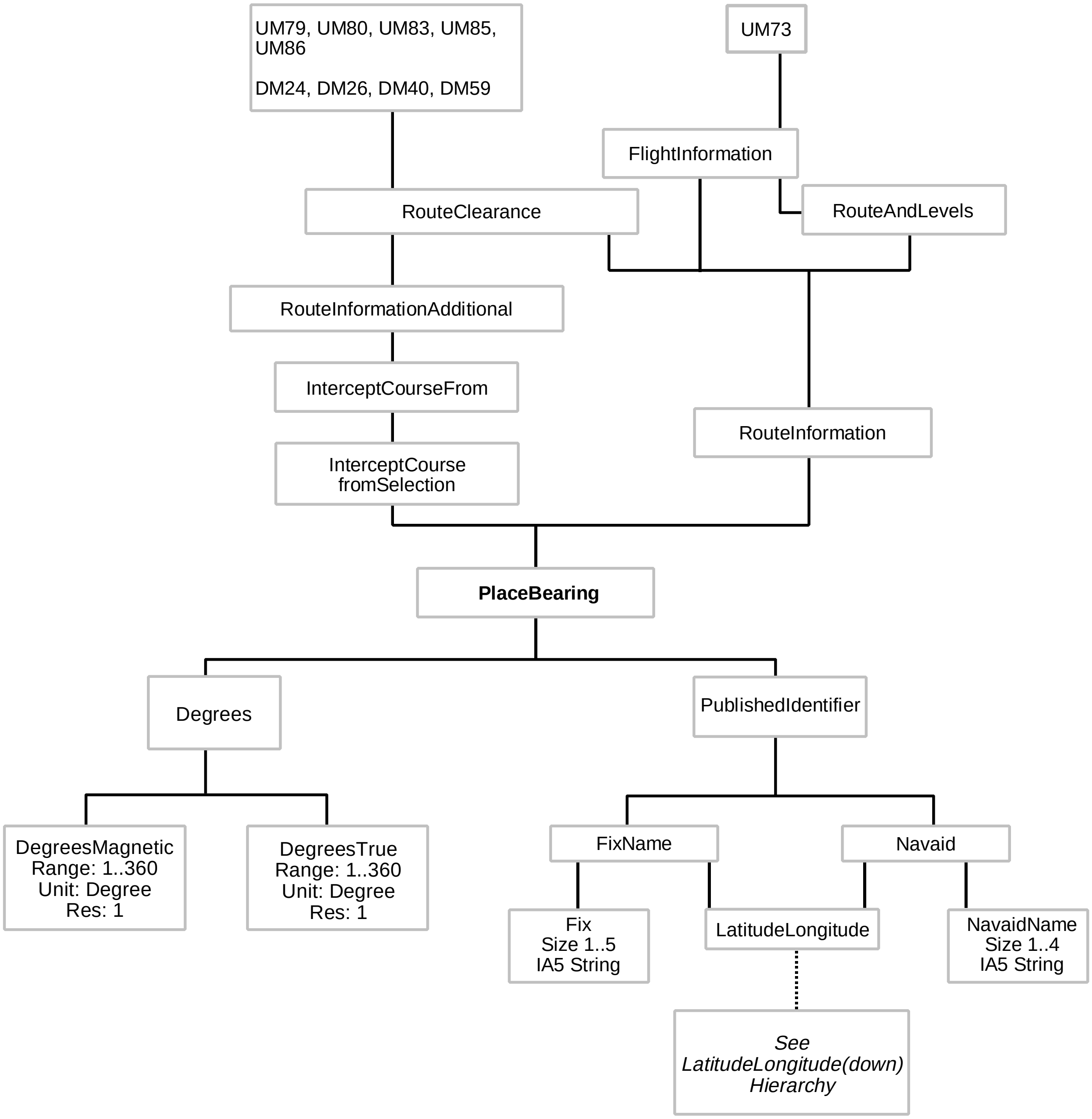

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
UM73
FlightInformation
RouteClearance
RouteAndLevels
RouteInformationAdditional
InterceptCourseFrom
InterceptCourse fromSelection
RouteInformation
PlaceBearing
Degrees
PublishedIdentifier
DegreesMagnetic Range: 1..360 Unit: Degree Res: 1
DegreesTrue Range: 1..360 Unit: Degree Res: 1
FixName
Navaid
Fix Size 1..5 IA5 String
LatitudeLongitude
NavaidName Size 1..4 IA5 String
See LatitudeLongitude(down) Hierarchy

Figure 4.6-92. PlaceBearing Variable Hierarchy

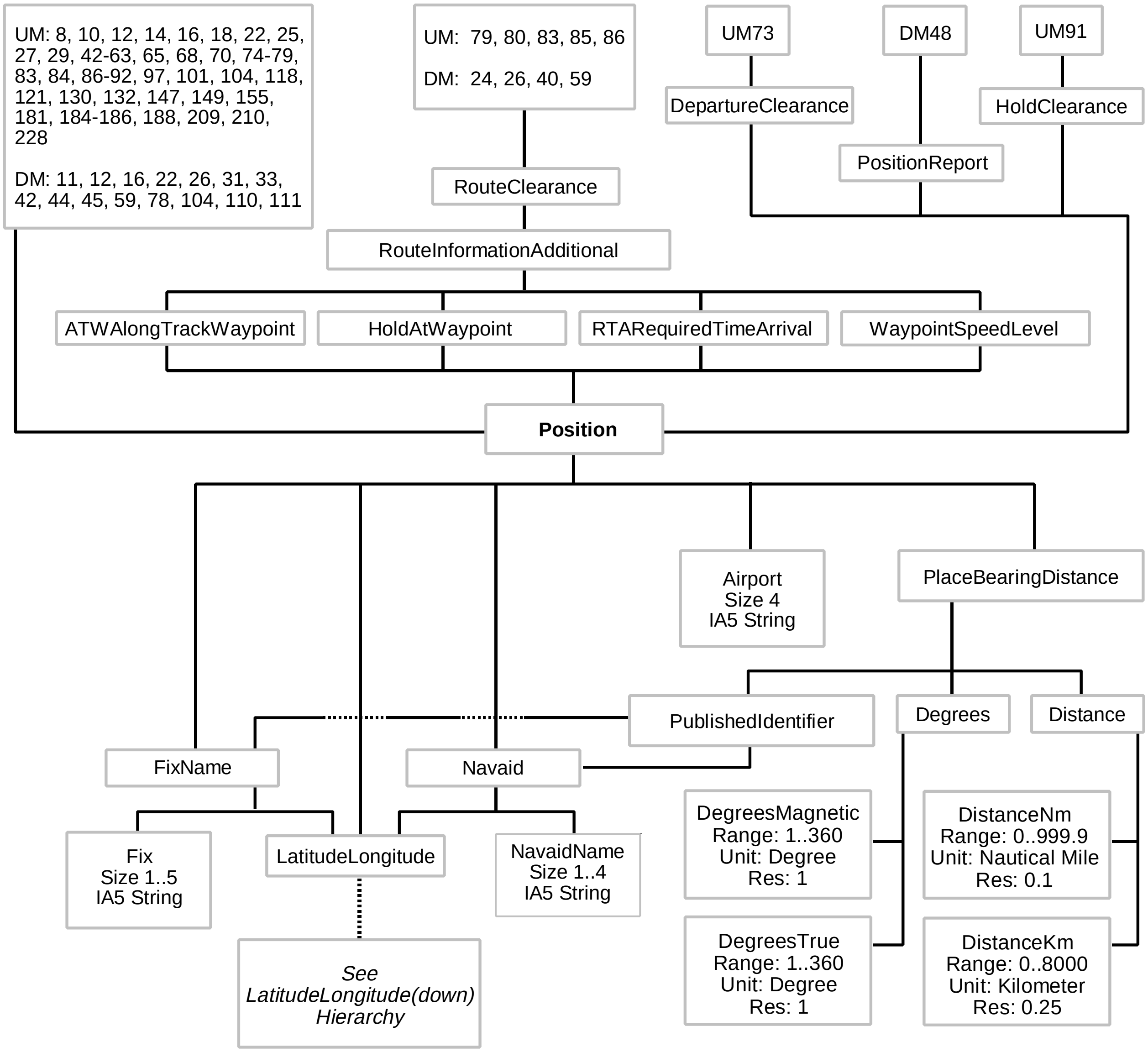


Figure 4.6-93. Position Variable Hierarchy

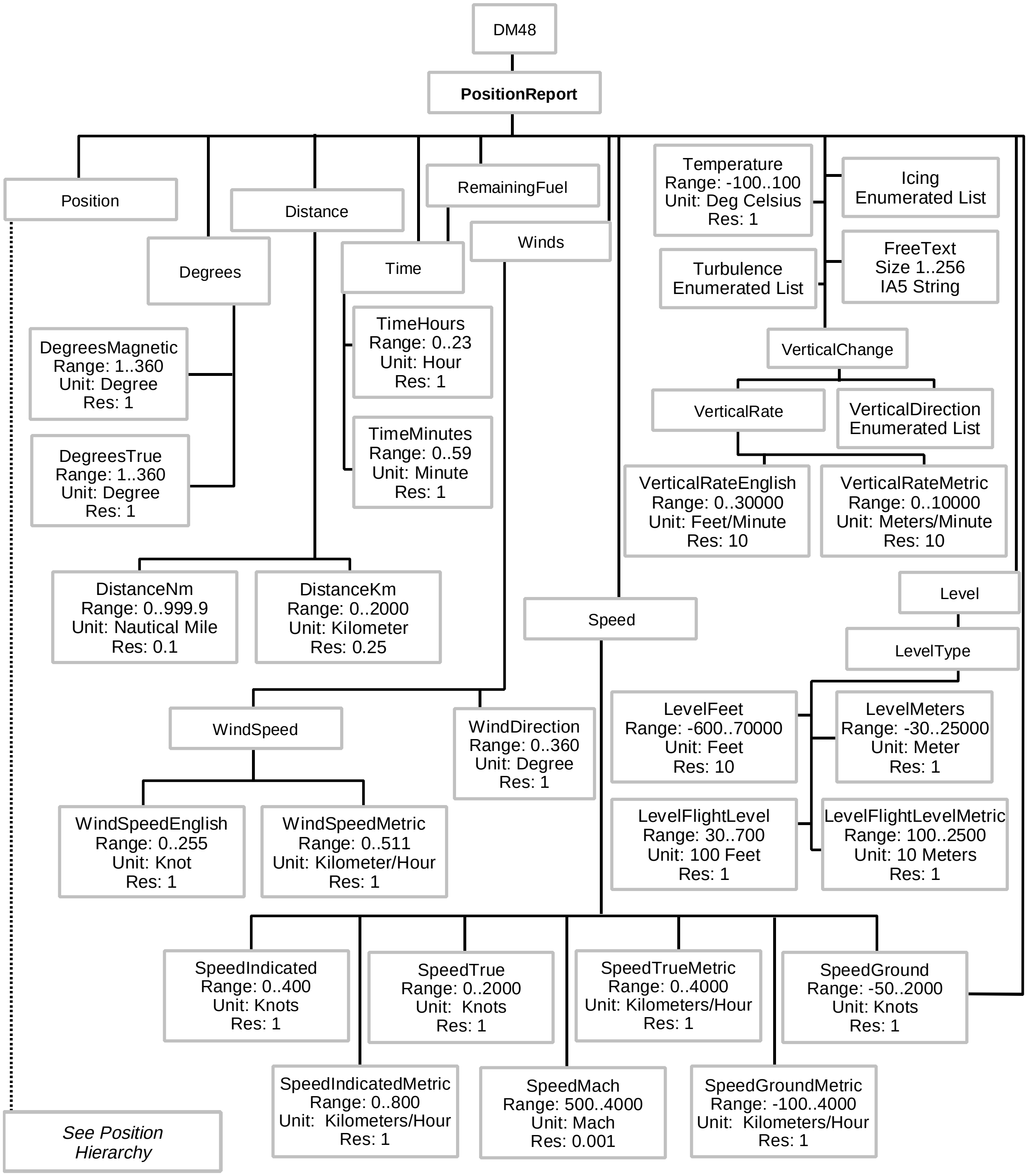

DM48
PositionReport
Position
Distance
RemainingFuel
Winds
Degrees
Time
Temperature Range: -100..100 Unit: Deg Celsius Res: 1
Icing Enumerated List
Turbulence Enumerated List
FreeText Size 1..256 IA5 String
VerticalChange
VerticalRate
VerticalDirection Enumerated List
DegreesMagnetic Range: 1..360 Unit: Degree Res: 1
DegreesTrue Range: 1..360 Unit: Degree Res: 1
TimeHours Range: 0..23 Unit: Hour Res: 1
TimeMinutes Range: 0..59 Unit: Minute Res: 1
VerticalRateEnglish Range: 0..30000 Unit: Feet/Minute Res: 10
VerticalRateMetric Range: 0..10000 Unit: Meters/Minute Res: 10
DistanceNm Range: 0..999.9 Unit: Nautical Mile Res: 0.1
DistanceKm Range: 0..2000 Unit: Kilometer Res: 0.25
Speed
Level
LevelType
WindSpeed
WindDirection Range: 0..360 Unit: Degree Res: 1
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
WindSpeedEnglish Range: 0..255 Unit: Knot Res: 1
WindSpeedMetric Range: 0..511 Unit: Kilometer/Hour Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
SpeedIndicated Range: 0..400 Unit: Knots Res: 1
SpeedTrue Range: 0..2000 Unit: Knots Res: 1
SpeedTrueMetric Range: 0..4000 Unit: Kilometers/Hour Res: 1
SpeedGround Range: -50..2000 Unit: Knots Res: 1
SpeedIndicatedMetric Range: 0..800 Unit: Kilometers/Hour Res: 1
SpeedMach Range: 500..4000 Unit: Mach Res: 0.001
SpeedGroundMetric Range: -100..4000 Unit: Kilometers/Hour Res: 1
See Position Hierarchy

Figure 4.6-94. PositionReport Variable Hierarchy

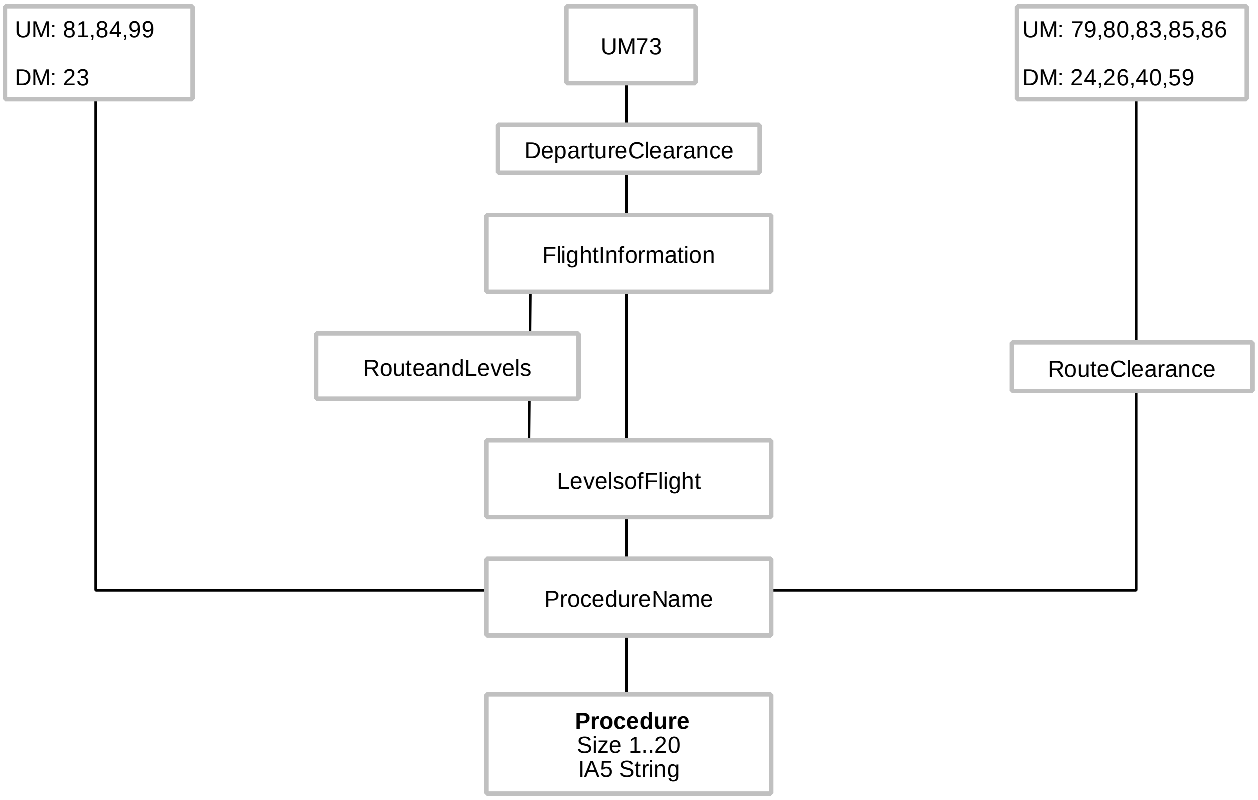


Figure 4.6-95. Procedure Variable Hierarchy

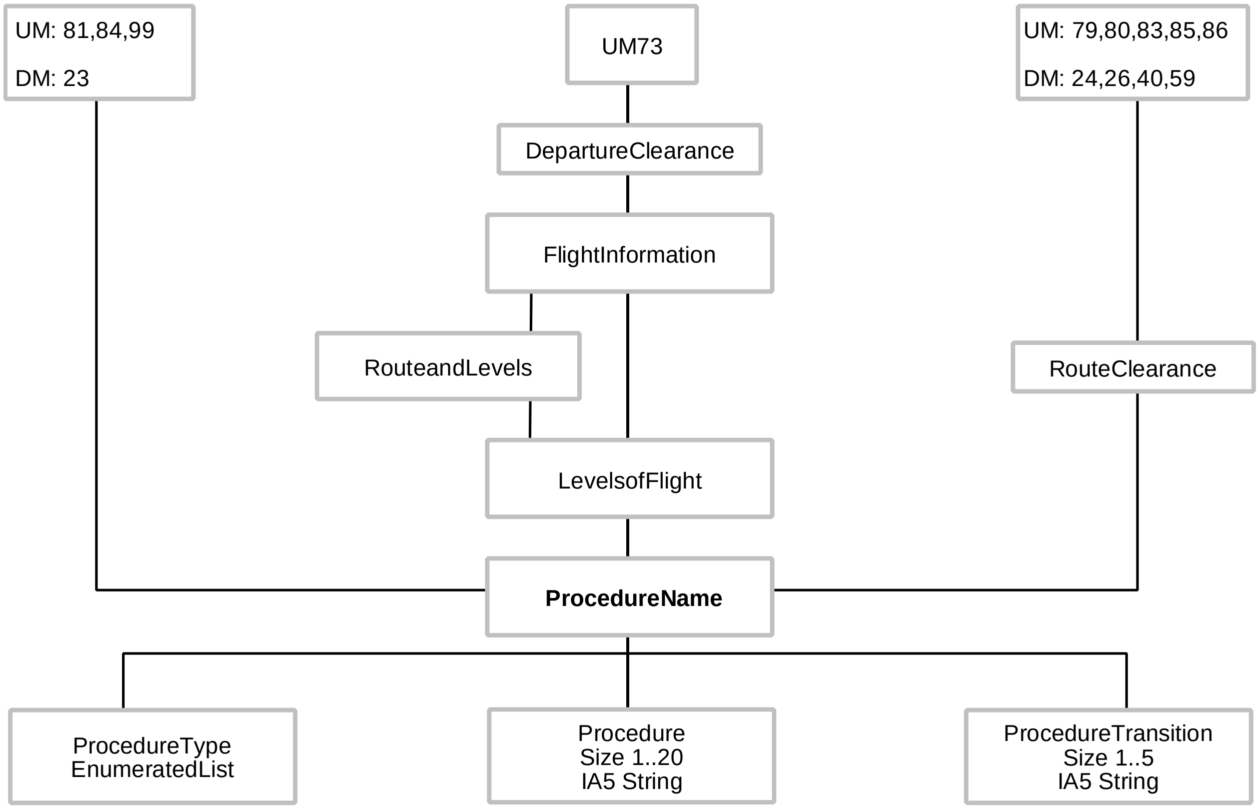


Figure 4.6-96. ProcedureName Variable Hierarchy

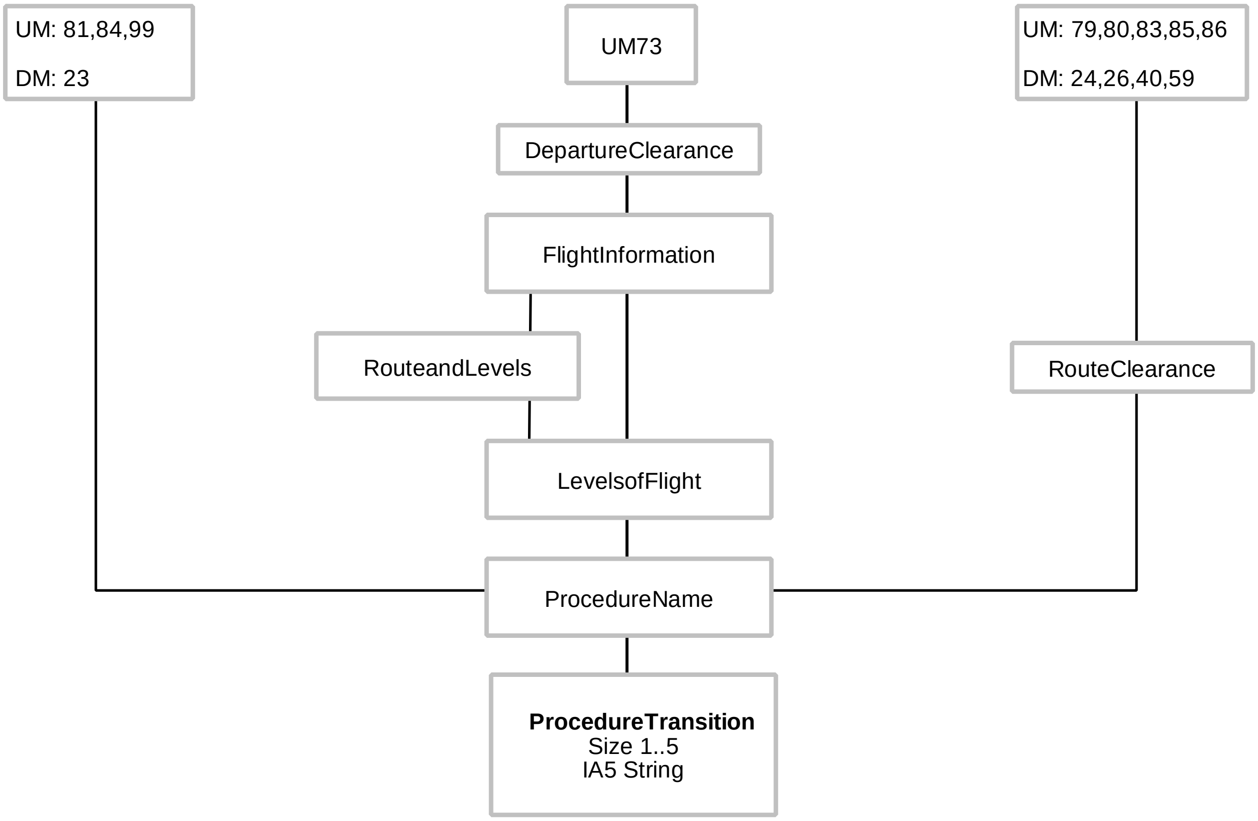


Figure 4.6-97. ProcedureTransition Variable Hierarchy

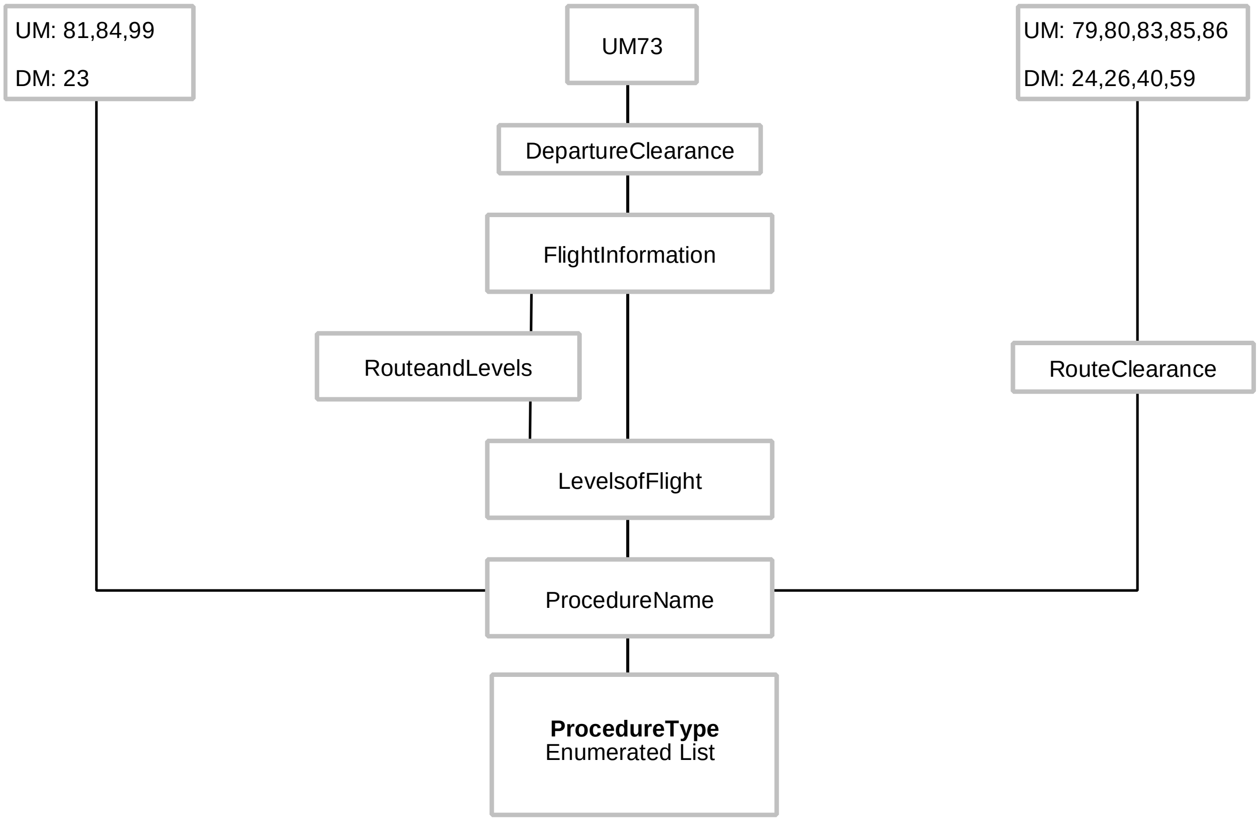


Figure 4.6-98. ProcedureType Variable Hierarchy

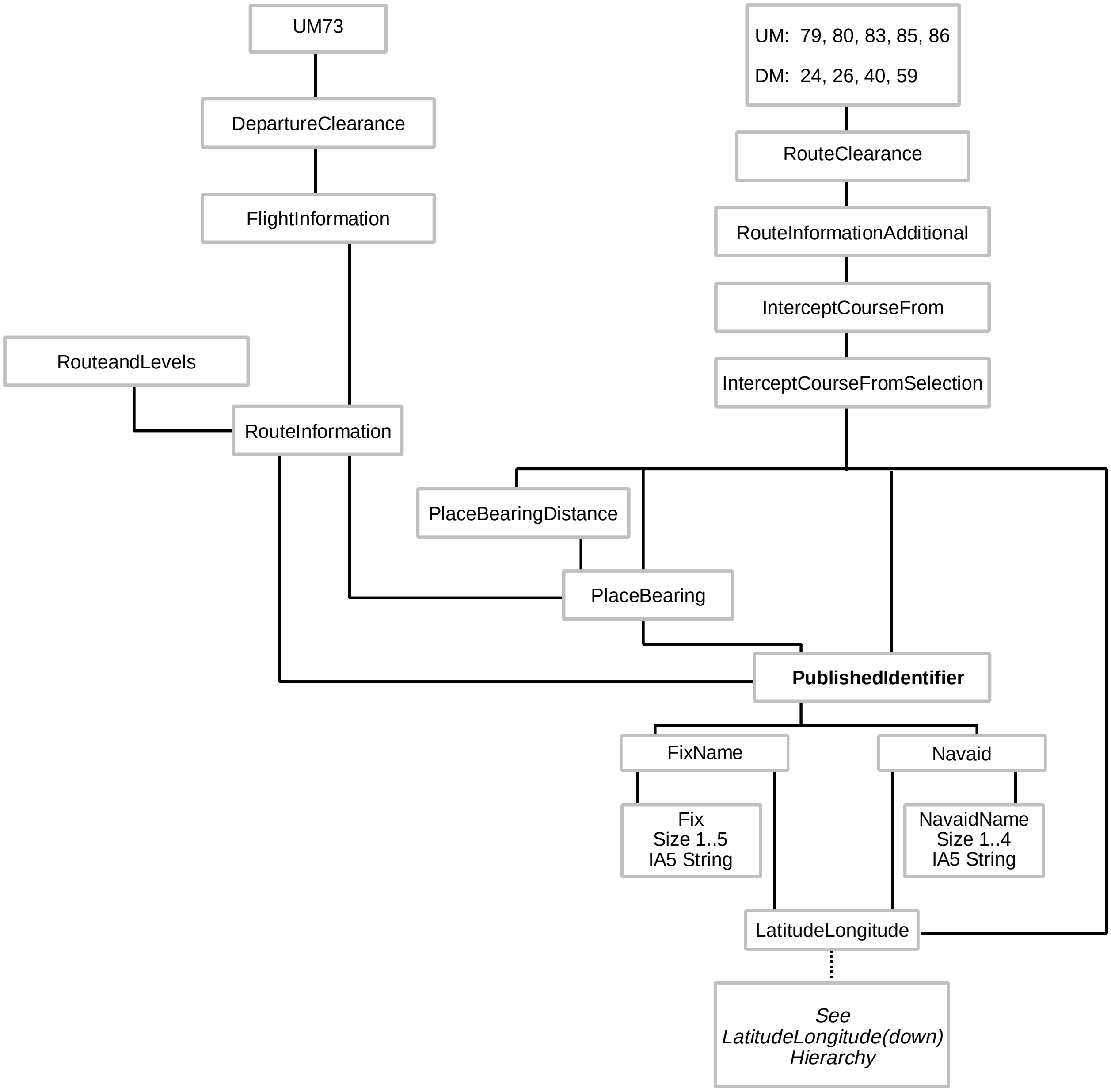


Figure 4.6-99. PublishedIdentifier Variable Hierarchy

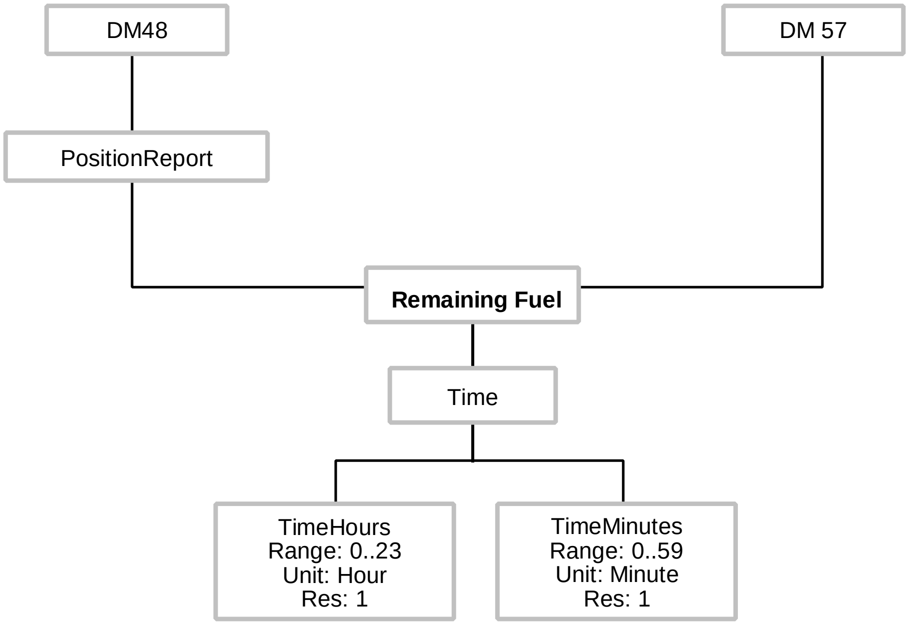


Figure 4.6-100. RemainingFuel Variable Hierarchy

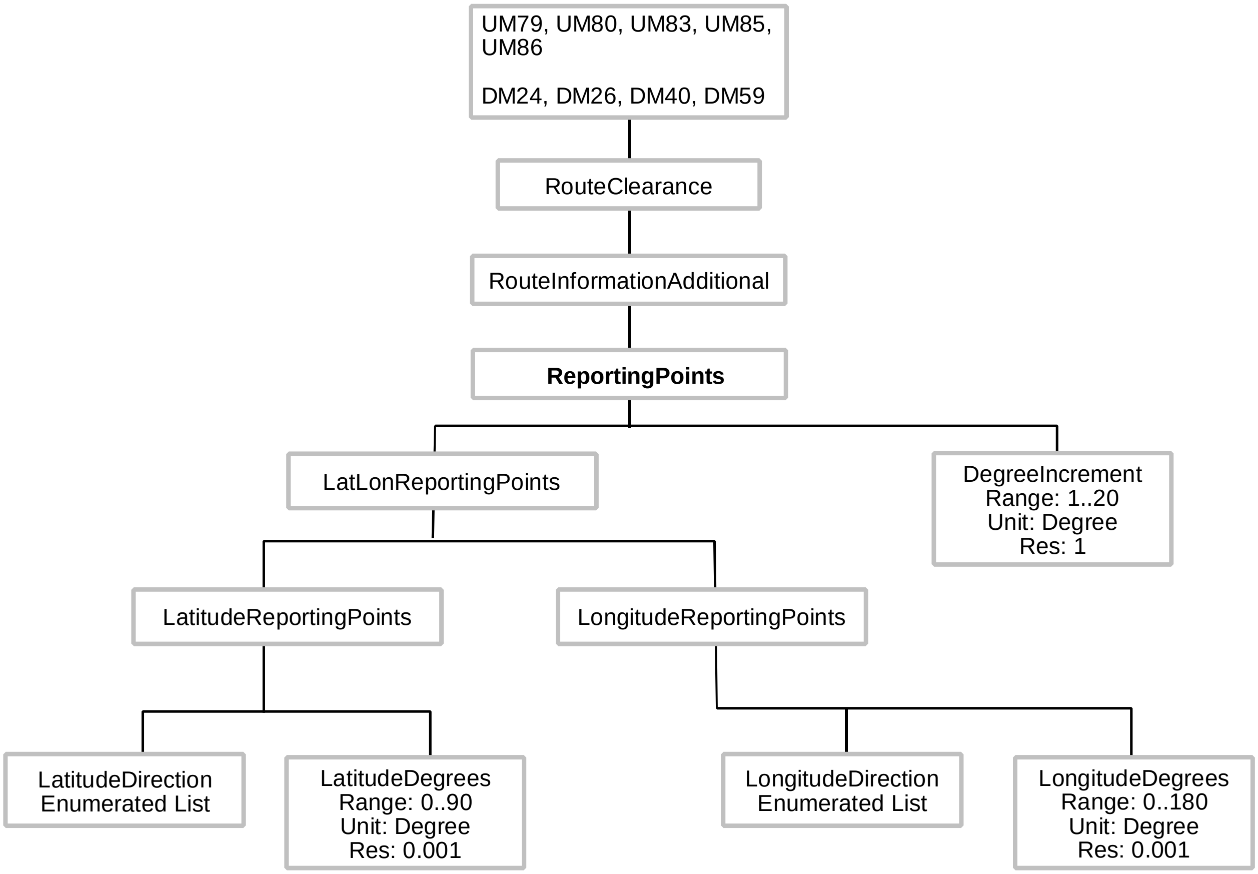

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
ReportingPoints
LatLonReportingPoints
DegreeIncrement Range: 1..20 Unit: Degree Res: 1
LatitudeReportingPoints
LongitudeReportingPoints
LatitudeDirection Enumerated List
LatitudeDegrees Range: 0..90 Unit: Degree Res: 0.001
LongitudeDirection Enumerated List
LongitudeDegrees Range: 0..180 Unit: Degree Res: 0.001

Figure 4.6-101. ReportingPoints Variable Hierarchy

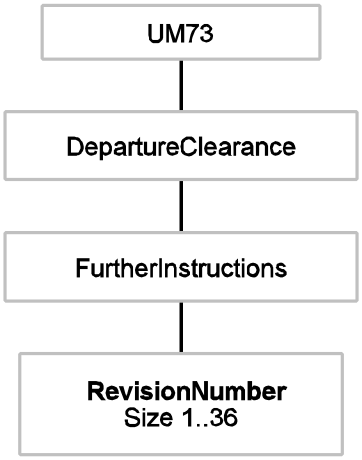


Figure 4.6-102. RevisionNumber Variable Hierarchy

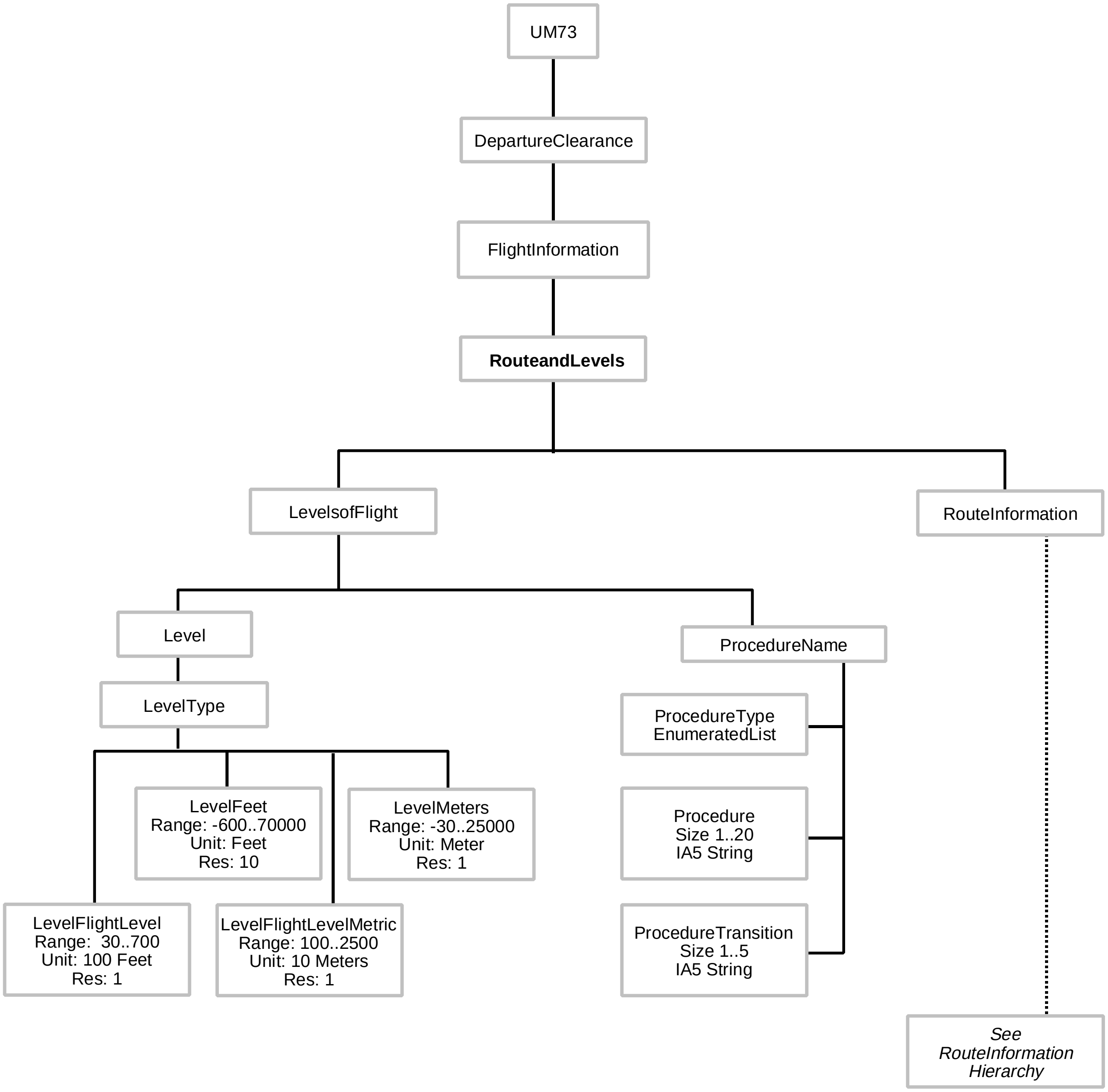

UM73
DepartureClearance
FlightInformation
RouteandLevels
LevelsofFlight
RouteInformation
Level
ProcedureName
LevelType
ProcedureType EnumeratedList
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
Procedure Size 1..20 IA5 String
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
ProcedureTransition Size 1..5 IA5 String
See RouteInformation Hierarchy

Figure 4.6-103. RouteandLevels Variable Hierarchy

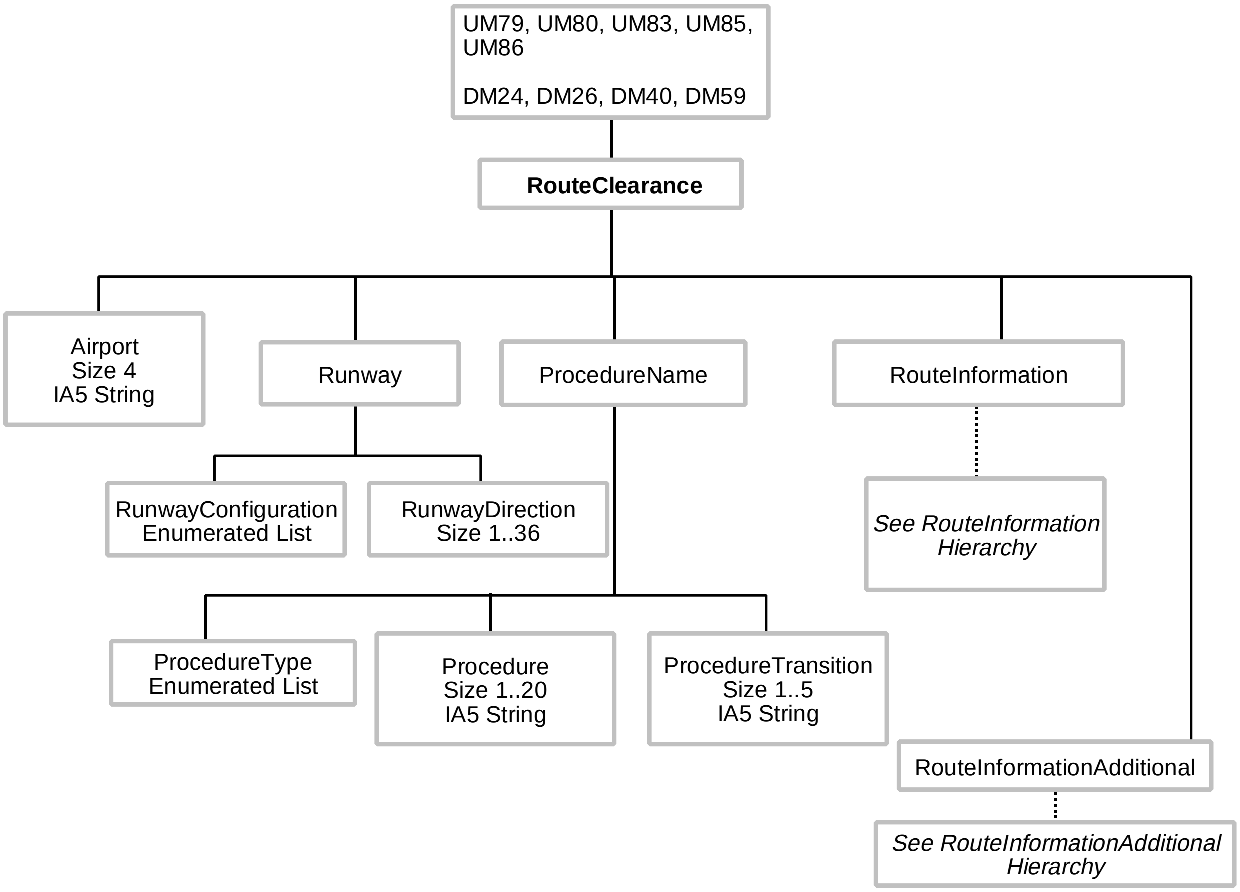

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
Airport Size 4 IA5 String
Runway
ProcedureName
RouteInformation
RunwayConfiguration Enumerated List
RunwayDirection Size 1..36
See RouteInformation Hierarchy
ProcedureType Enumerated List
Procedure Size 1..20 IA5 String
ProcedureTransition Size 1..5 IA5 String
RouteInformationAdditional
See RouteInformationAdditional Hierarchy

Figure 4.6-104. RouteClearance Variable Hierarchy

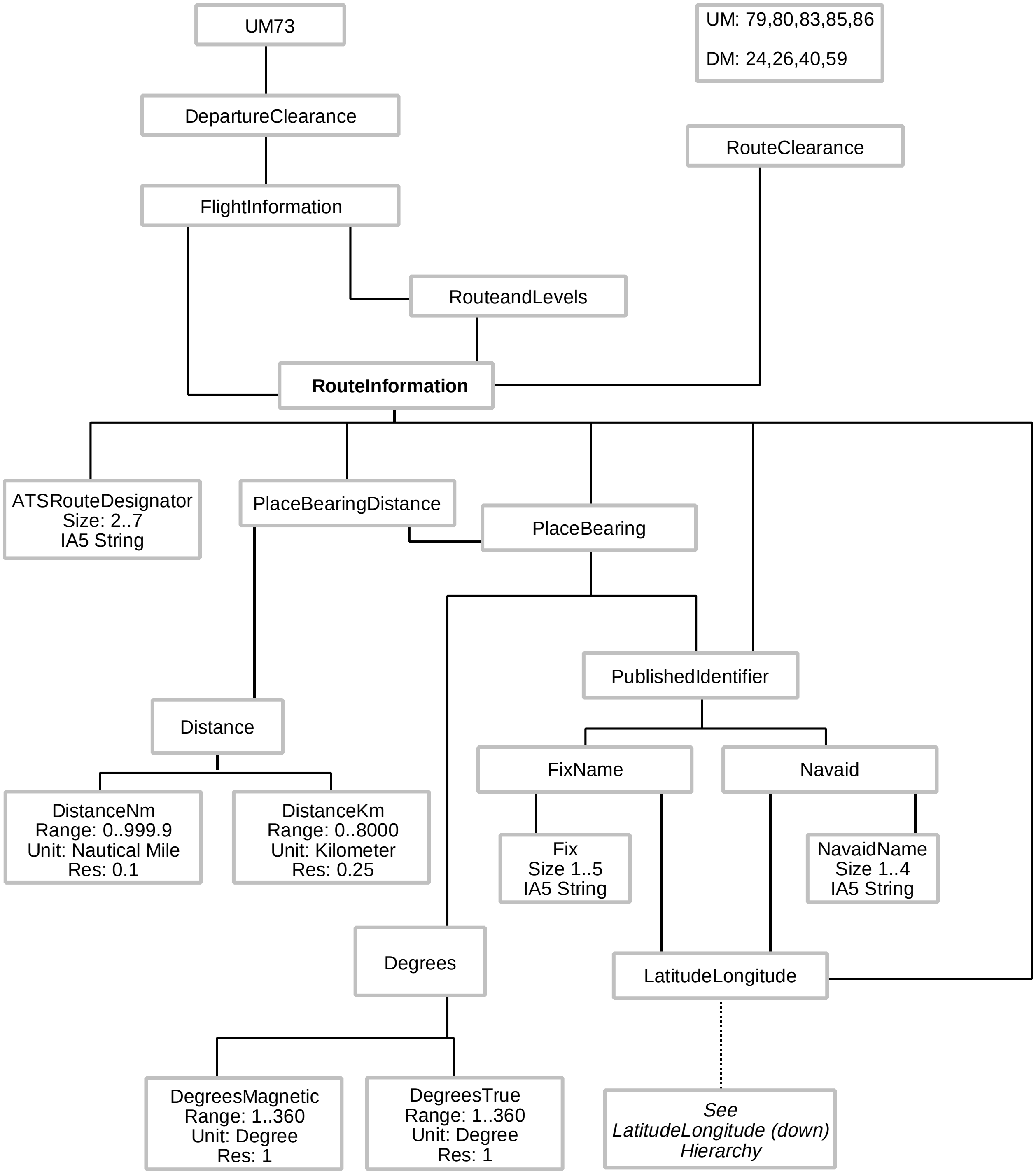


Figure 4.6-105. RouteInformation Variable Hierarchy

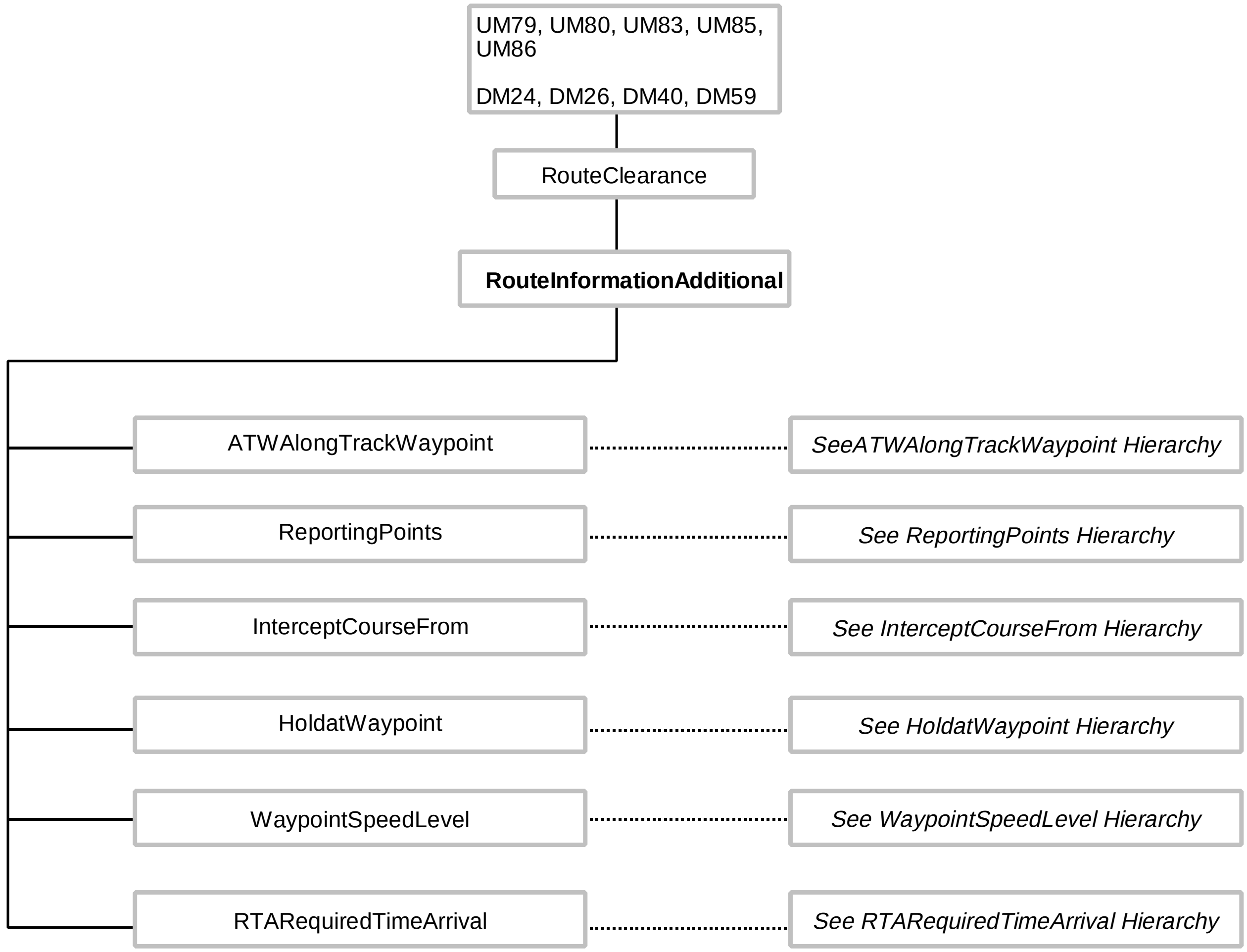

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
ATWAlongTrackWaypoint
SeeATWAlongTrackWaypoint Hierarchy
ReportingPoints
See ReportingPoints Hierarchy
InterceptCourseFrom
See InterceptCourseFrom Hierarchy
HoldatWaypoint
See HoldatWaypoint Hierarchy
WaypointSpeedLevel
See WaypointSpeedLevel Hierarchy
RTARequiredTimeArrival
See RTARequiredTimeArrival Hierarchy

Figure 4.6-106. RouteInformationAdditional Variable Hierarchy

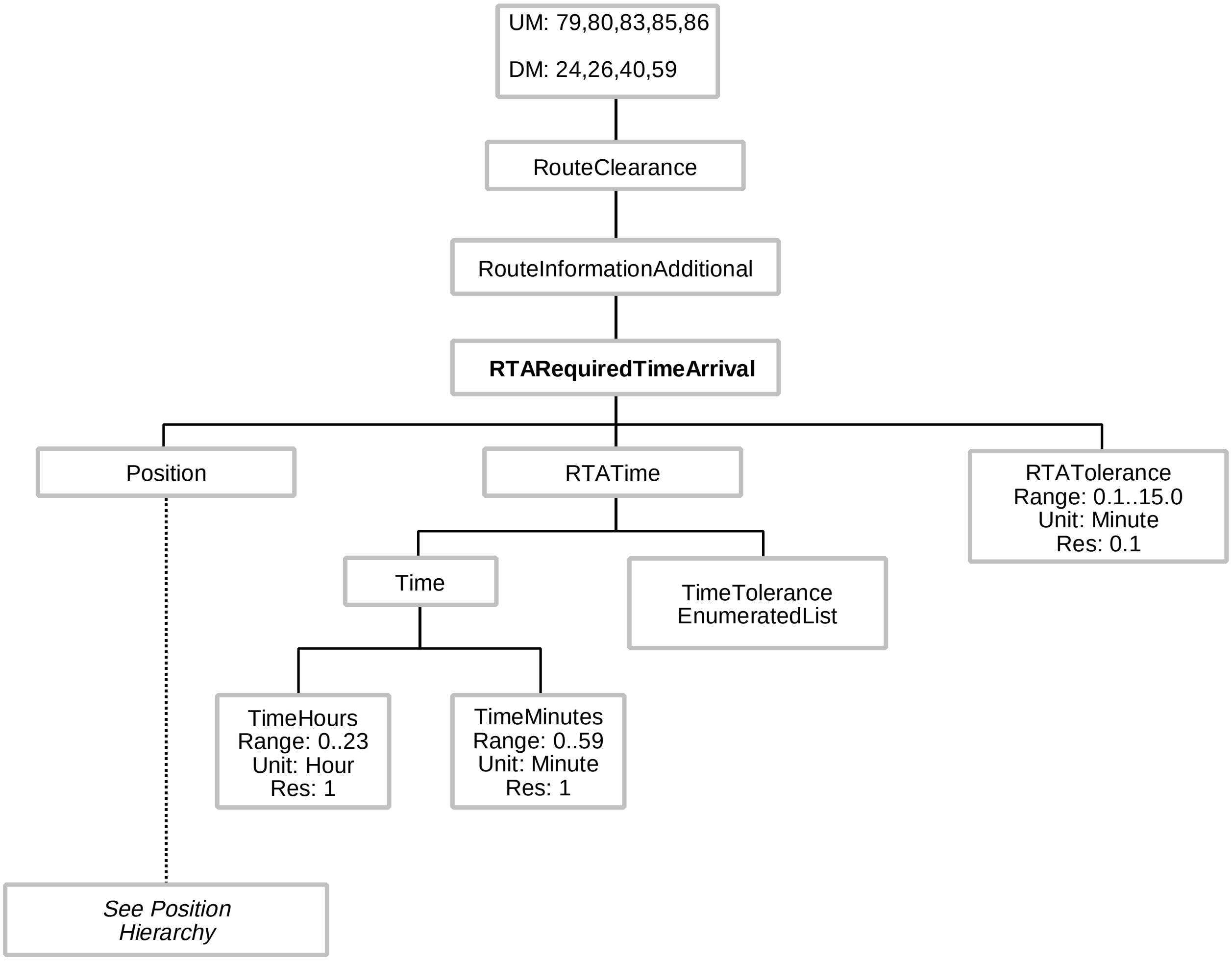


Figure 4.6-107. RTARequiredTimeArrival Variable Hierarchy

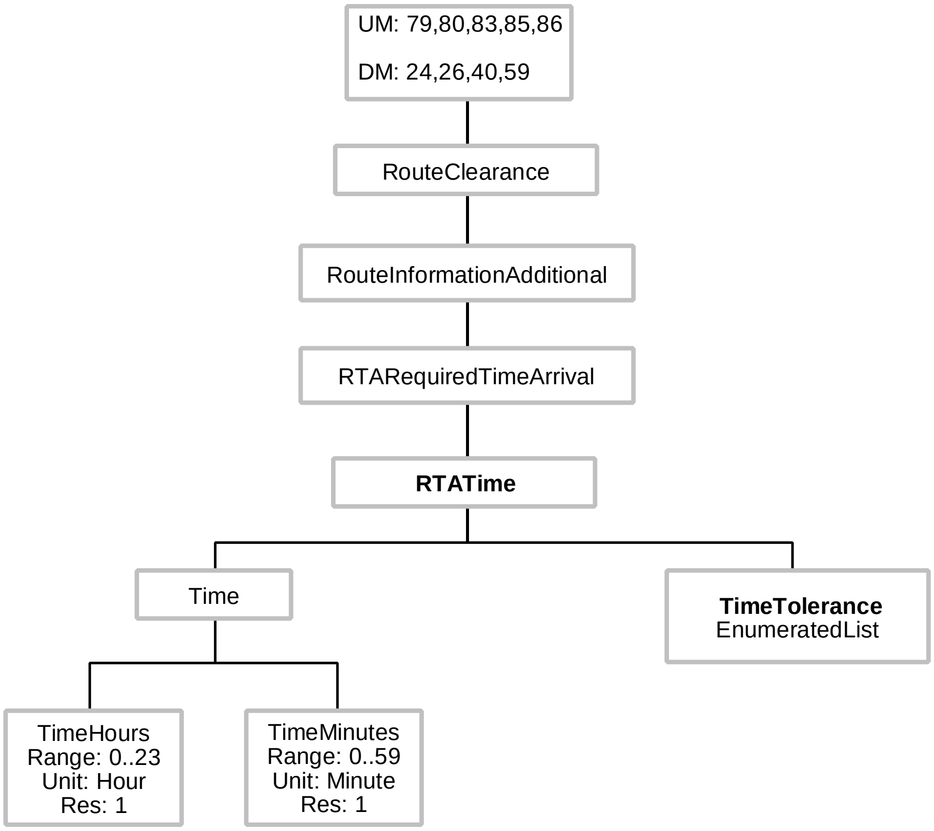


Figure 4.6-108. RTATime Variable Hierarchy

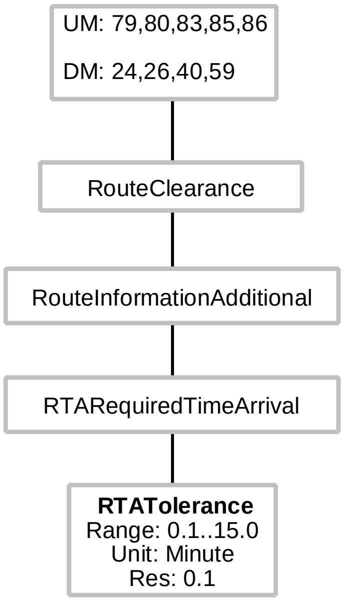


Figure 4.6-109. RTATolerance Variable Hierarchy

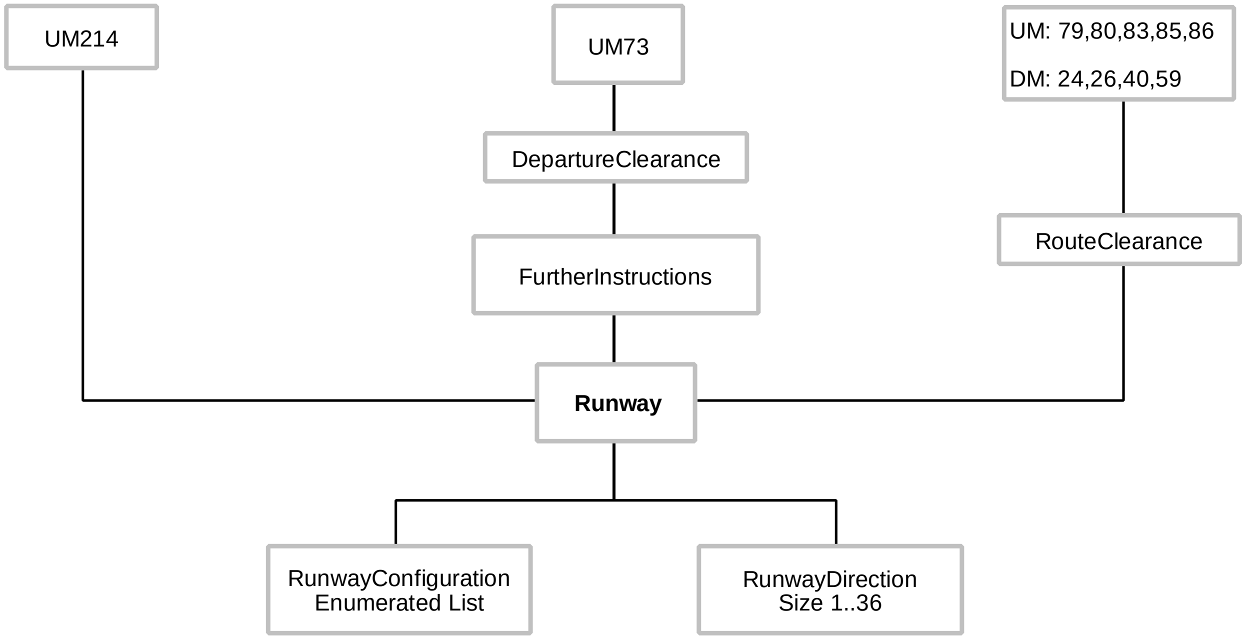


Figure 4.6-110. Runway Variable Hierarchy

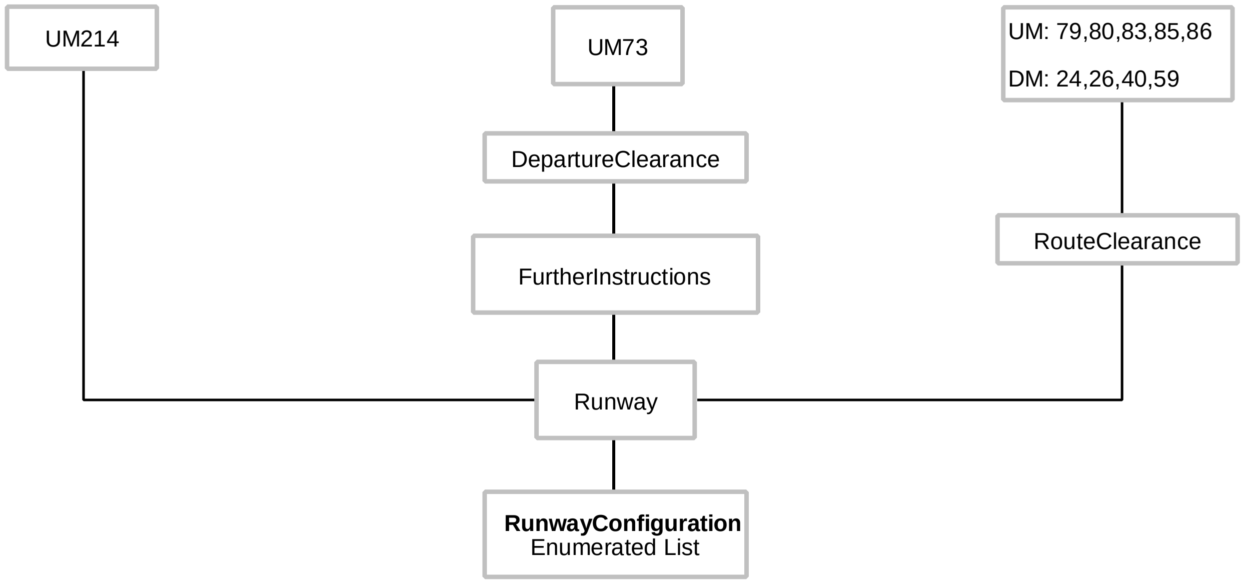


Figure 4.6-111. RunwayConfiguration Variable Hierarchy

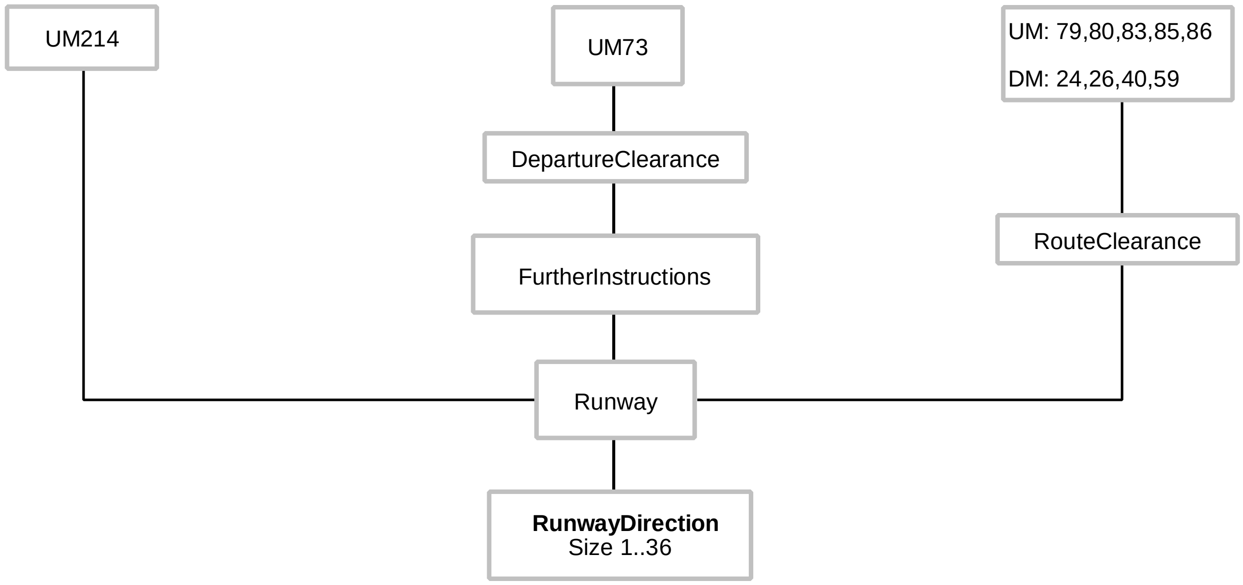


Figure 4.6-112. RunwayDirection Variable Hierarchy

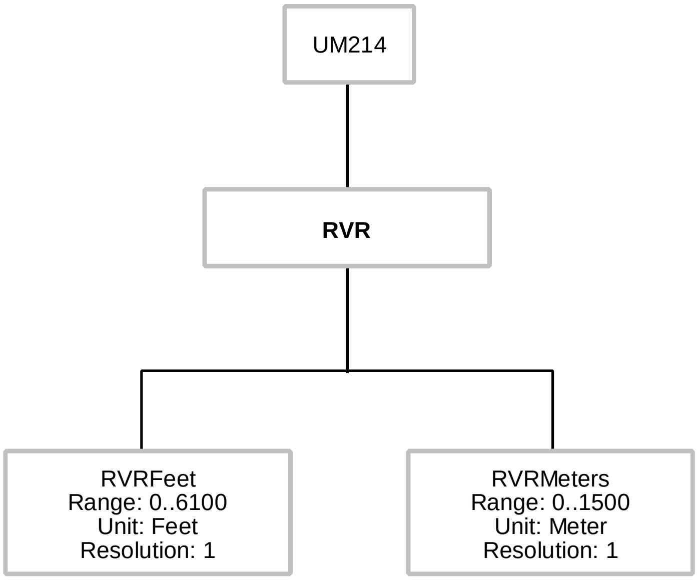


Figure 4.6-113. RVR Variable Hierarchy

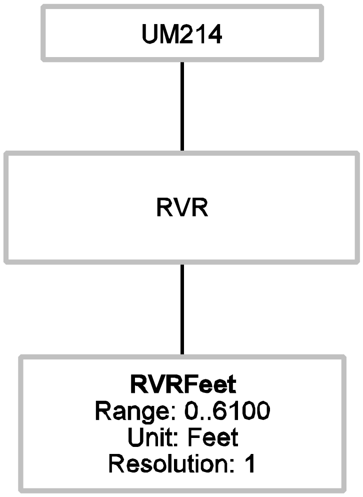


Figure 4.6-114. RVRFeet Variable Hierarchy

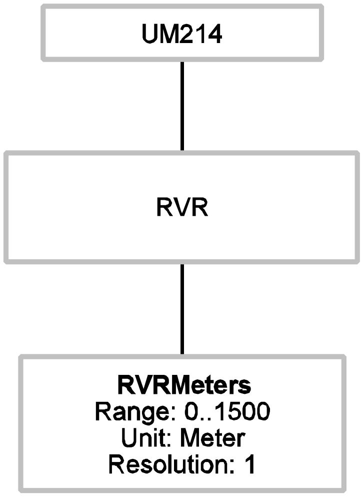


Figure 4.6-115. RVRMeters Variable Hierarchy

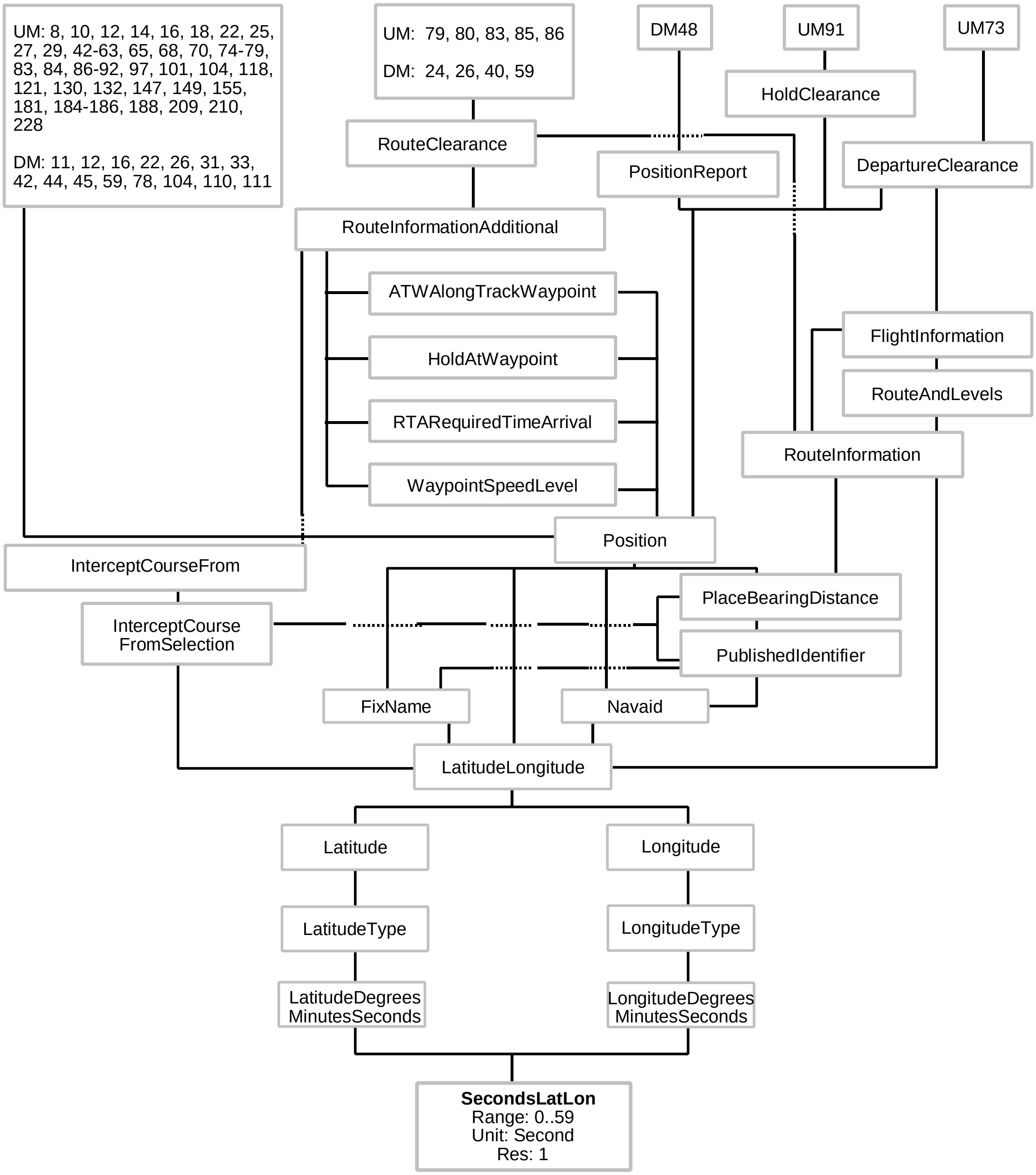


Figure 4.6-116. Seconds LatLon Variable Hierarchy

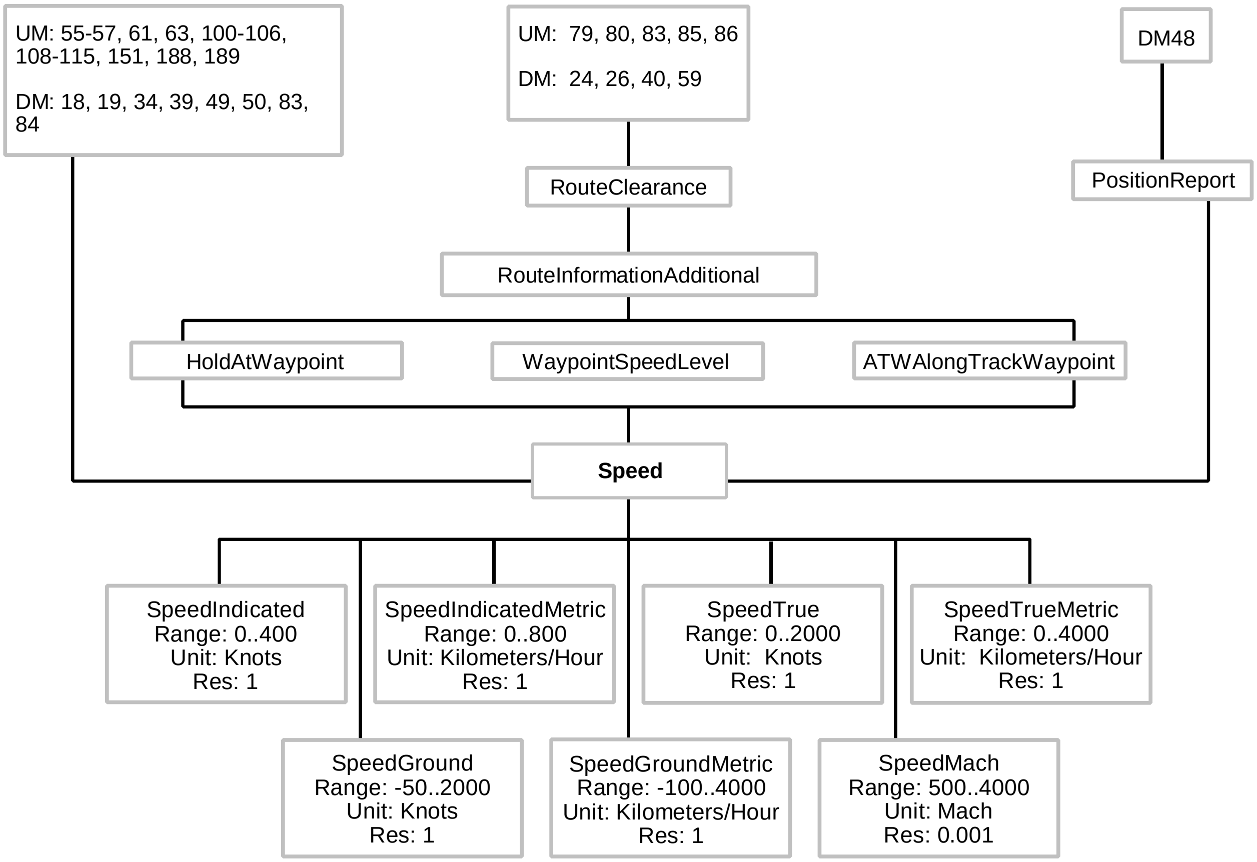

UM: 55-57, 61, 63, 100-106, 108-115, 151, 188, 189
DM: 18, 19, 34, 39, 49, 50, 83, 84
UM: 79, 80, 83, 85, 86
DM: 24, 26, 40, 59
DM48
RouteClearance
PositionReport
RouteInformationAdditional
HoldAtWaypoint
WaypointSpeedLevel
ATWAlongTrackWaypoint
Speed
SpeedIndicated Range: 0..400 Unit: Knots Res: 1
SpeedIndicatedMetric Range: 0..800 Unit: Kilometers/Hour Res: 1
SpeedTrue Range: 0..2000 Unit: Knots Res: 1
SpeedTrueMetric Range: 0..4000 Unit: Kilometers/Hour Res: 1
SpeedGround Range: -50..2000 Unit: Knots Res: 1
SpeedGroundMetric Range: -100..4000 Unit: Kilometers/Hour Res: 1
SpeedMach Range: 500..4000 Unit: Mach Res: 0.001

Figure 4.6-117. Speed Variable Hierarchy

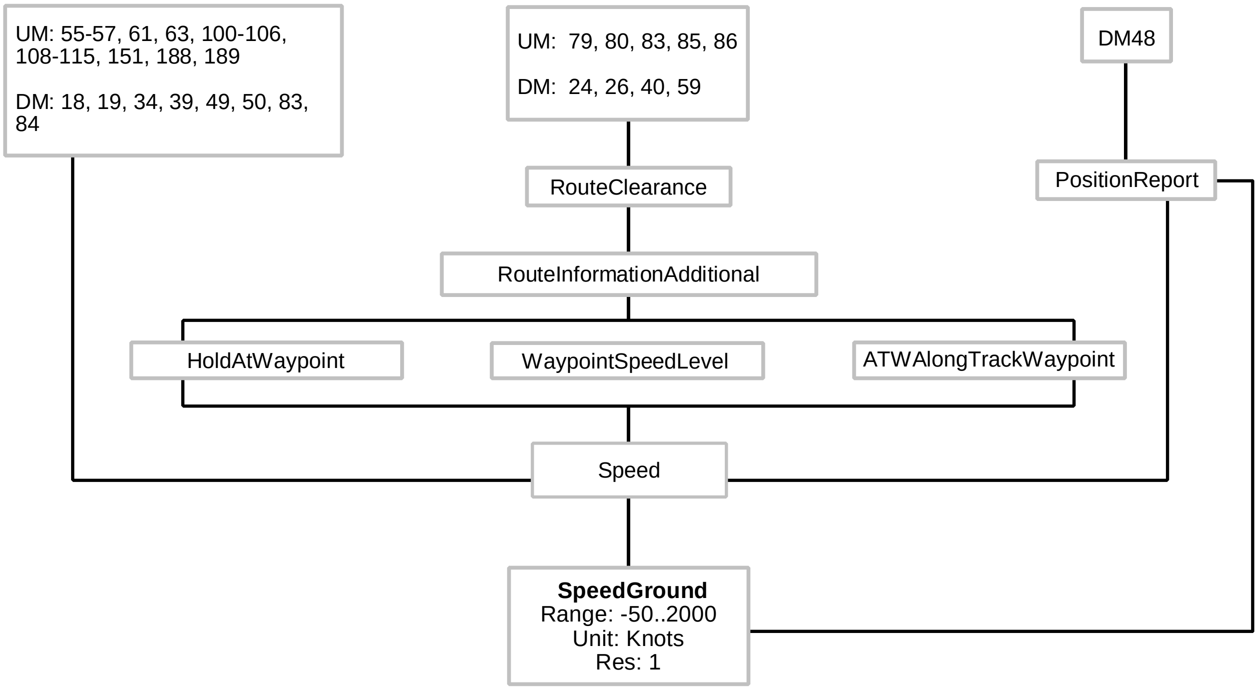


Figure 4.6-118. SpeedGround Variable Hierarchy

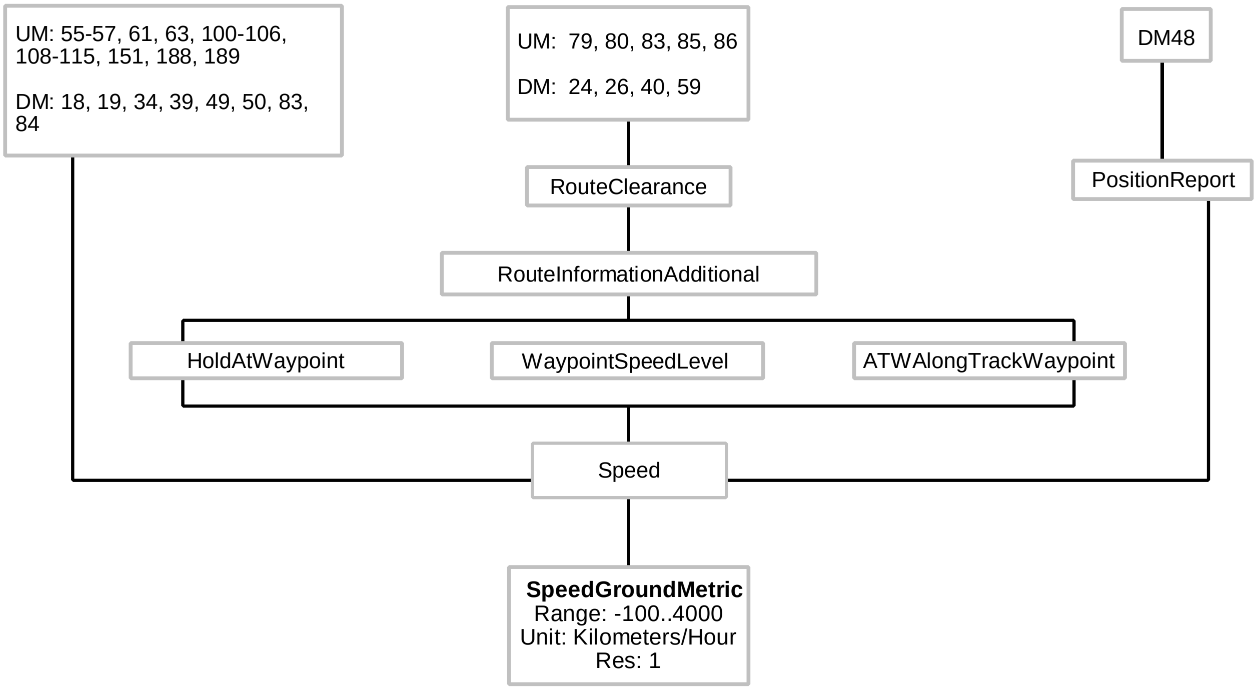


Figure 4.6-119. SpeedGroundMetric Variable Hierarchy

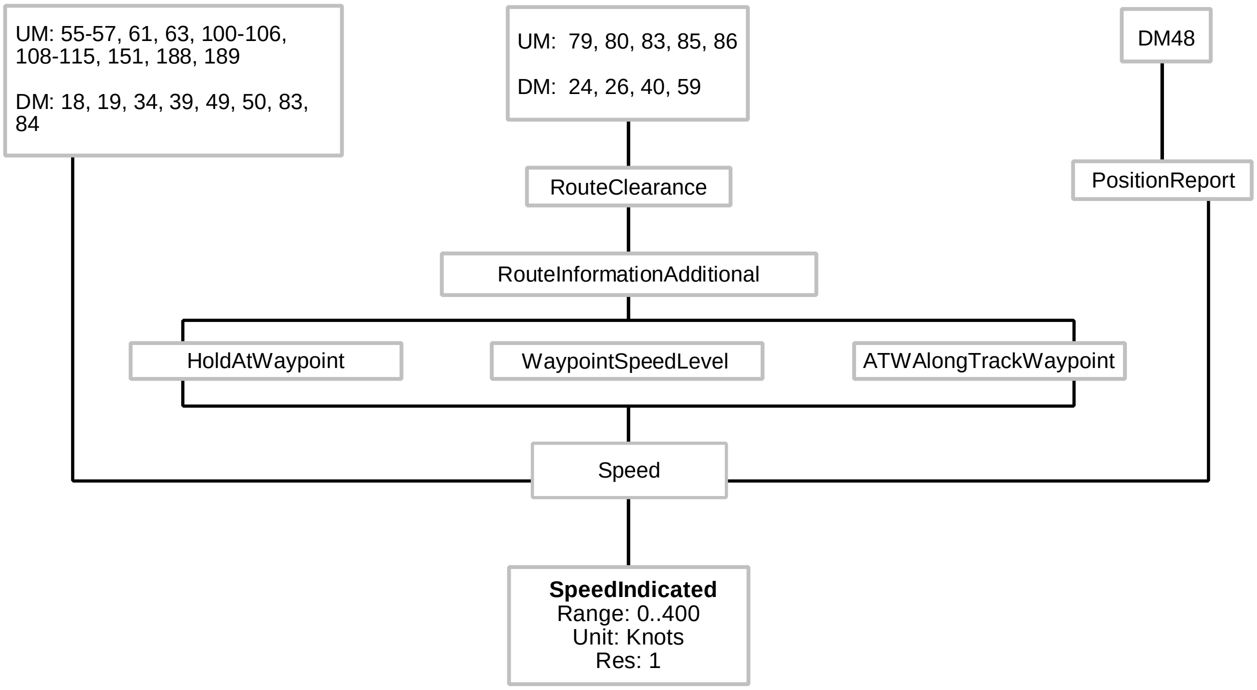


Figure 4.6-120. SpeedIndicated Variable Hierarchy

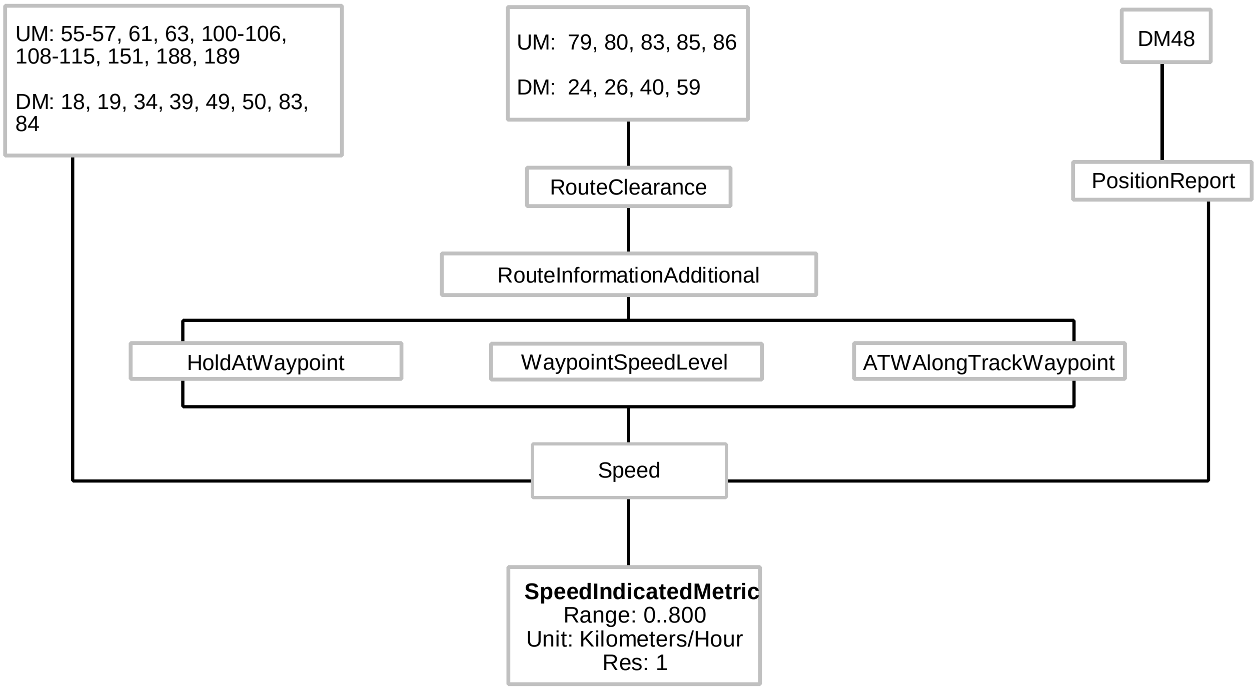


Figure 4.6-121. SpeedIndicatedMetric Variable Hierarchy

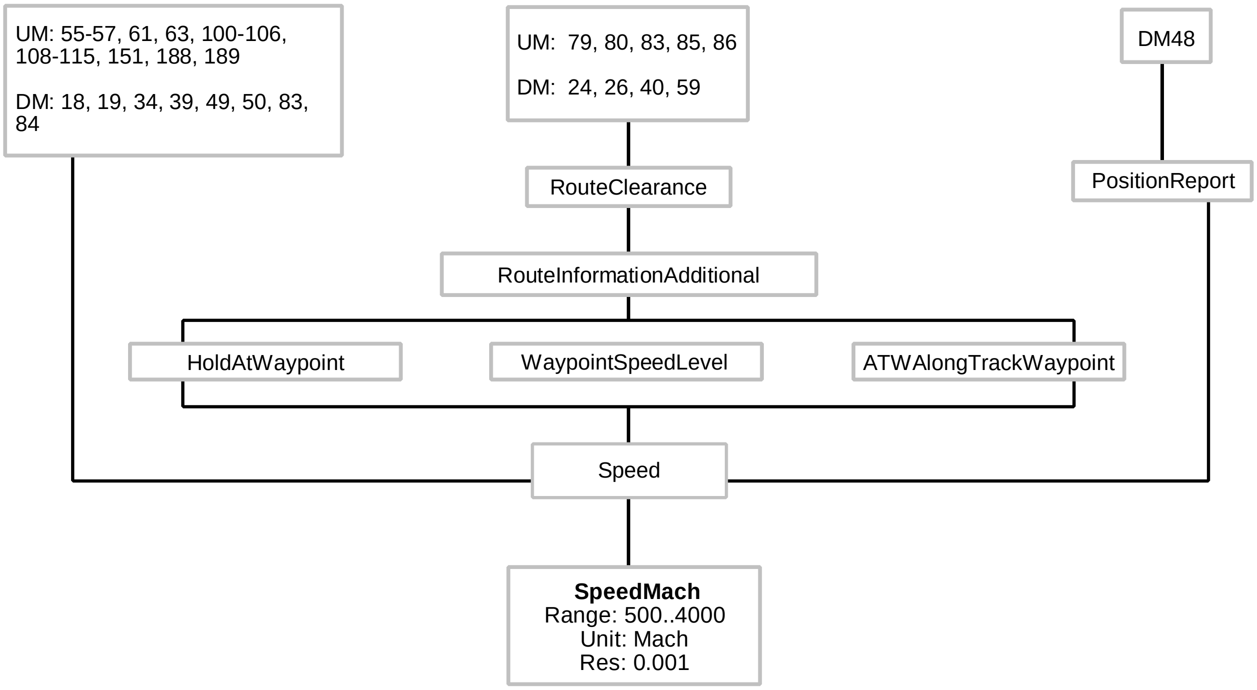


Figure 4.6-122. SpeedMach Variable Hierarchy

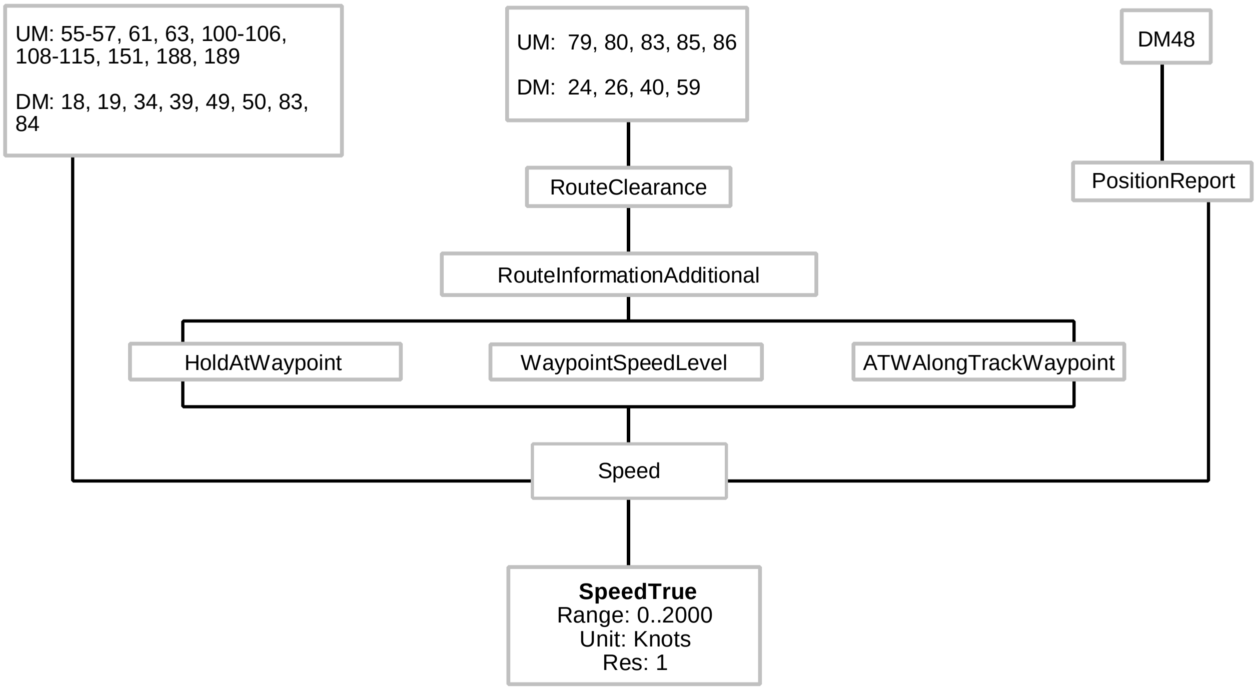


Figure 4.6-123. SpeedTrue Variable Hierarchy

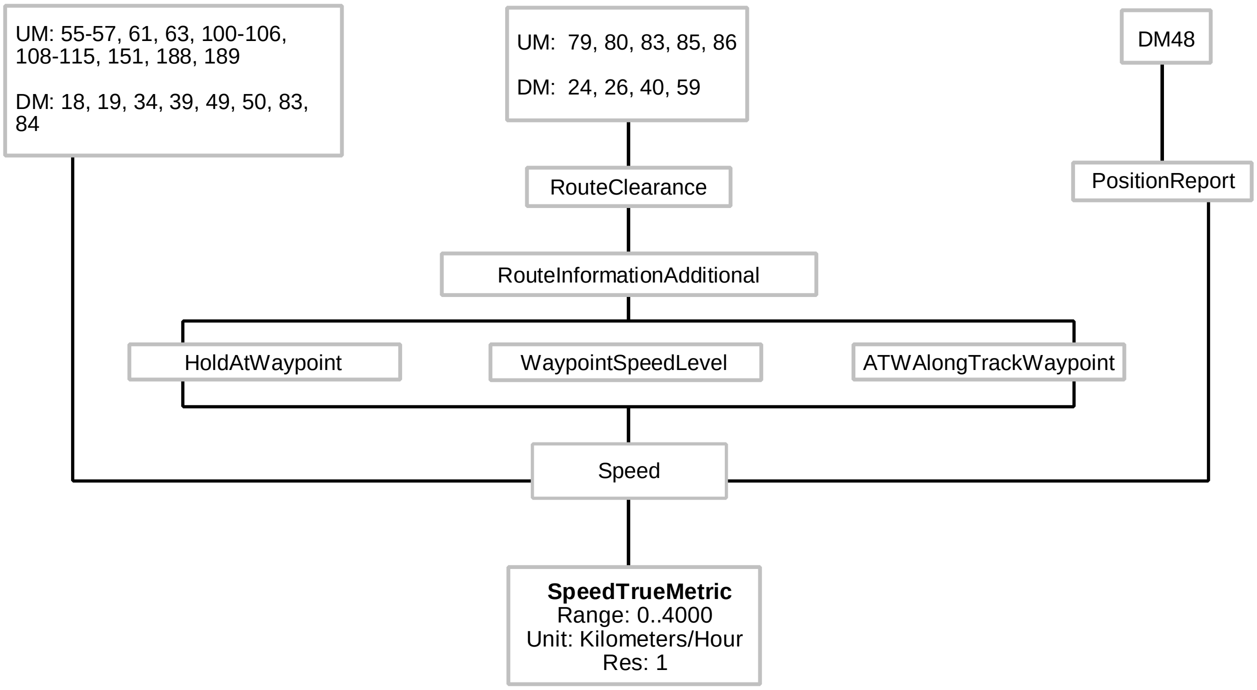


Figure 4.6-124. SpeedTrueMetric Variable Hierarchy

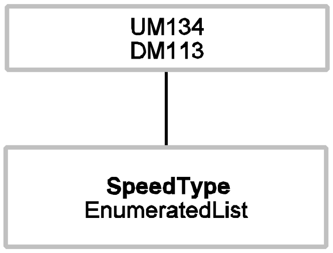


Figure 4.6-125. SpeedType Variable Hierarchy

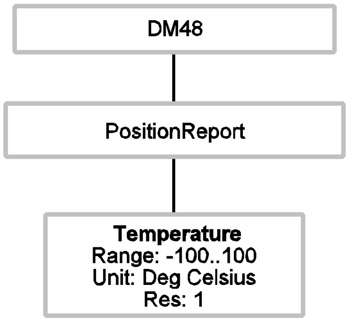


Figure 4.6-126. Temperature Variable Hierarchy

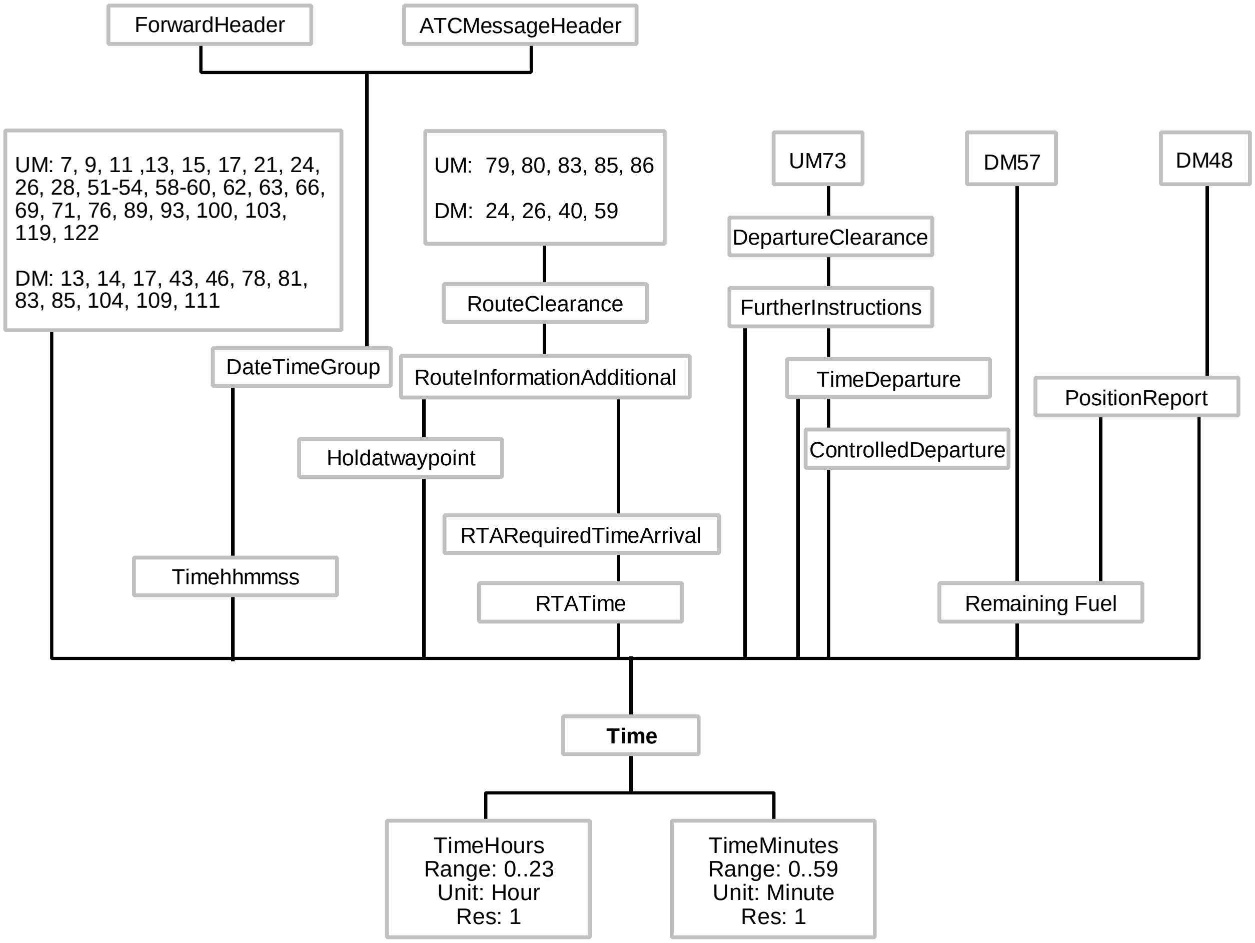


Figure 4.6-127. Time Variable Hierarchy

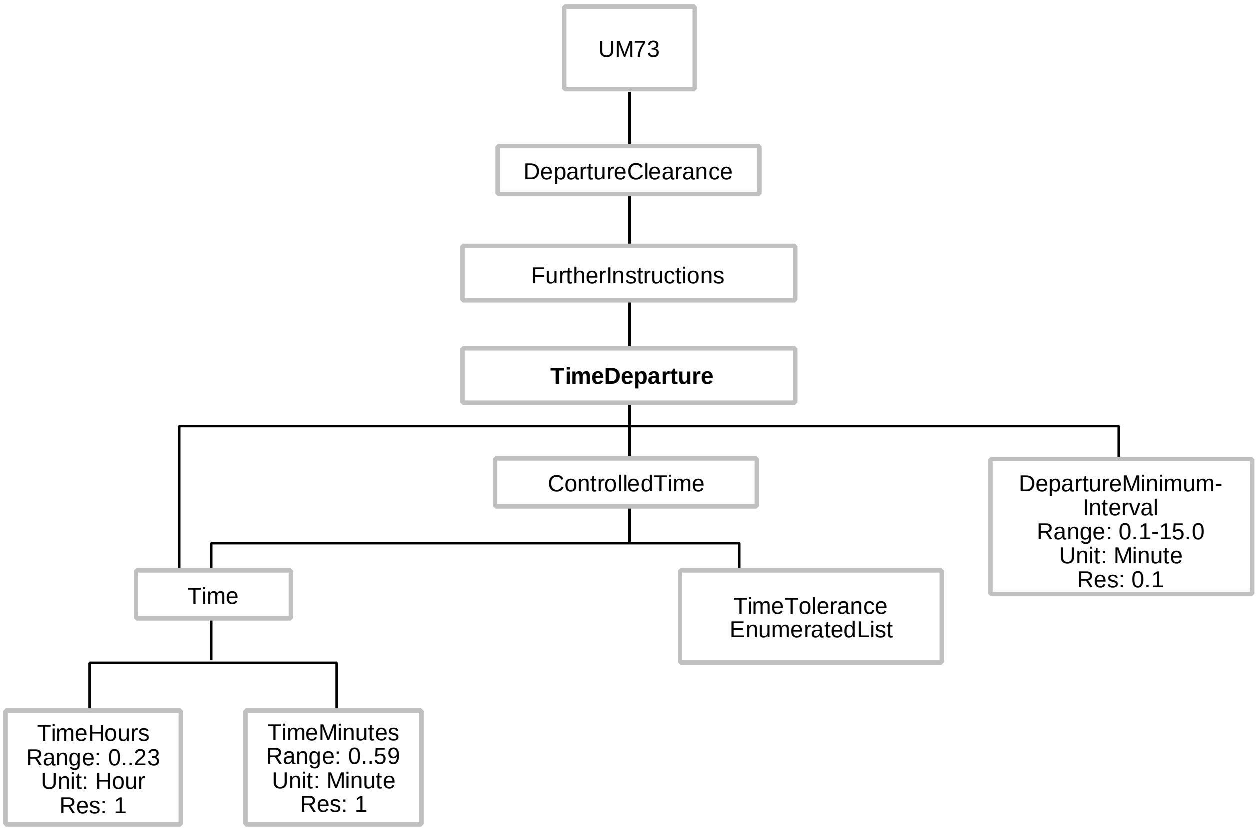


Figure 4.6-128. TimeDeparture Variable Hierarchy

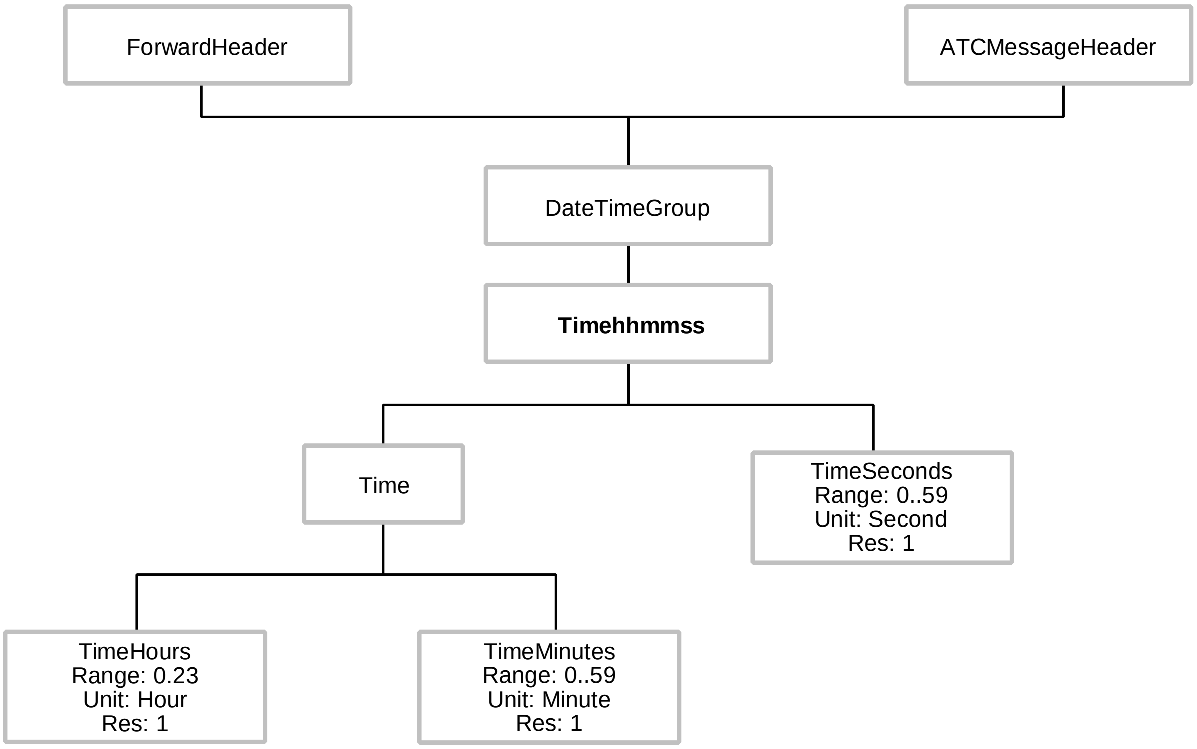


Figure 4.6-129. Timehhmmss Variable Hierarchy

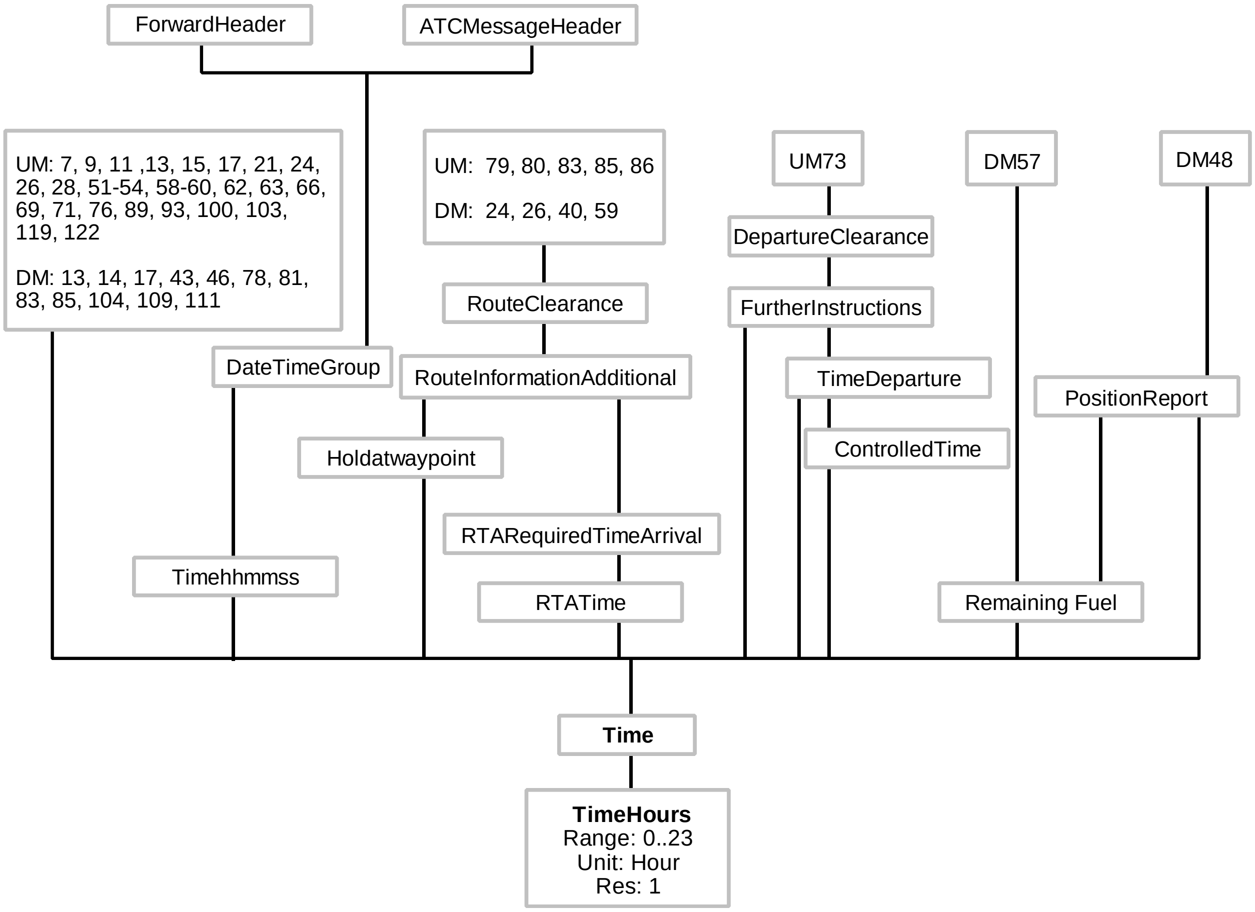


Figure 4.6-130. TimeHours Variable Hierarchy

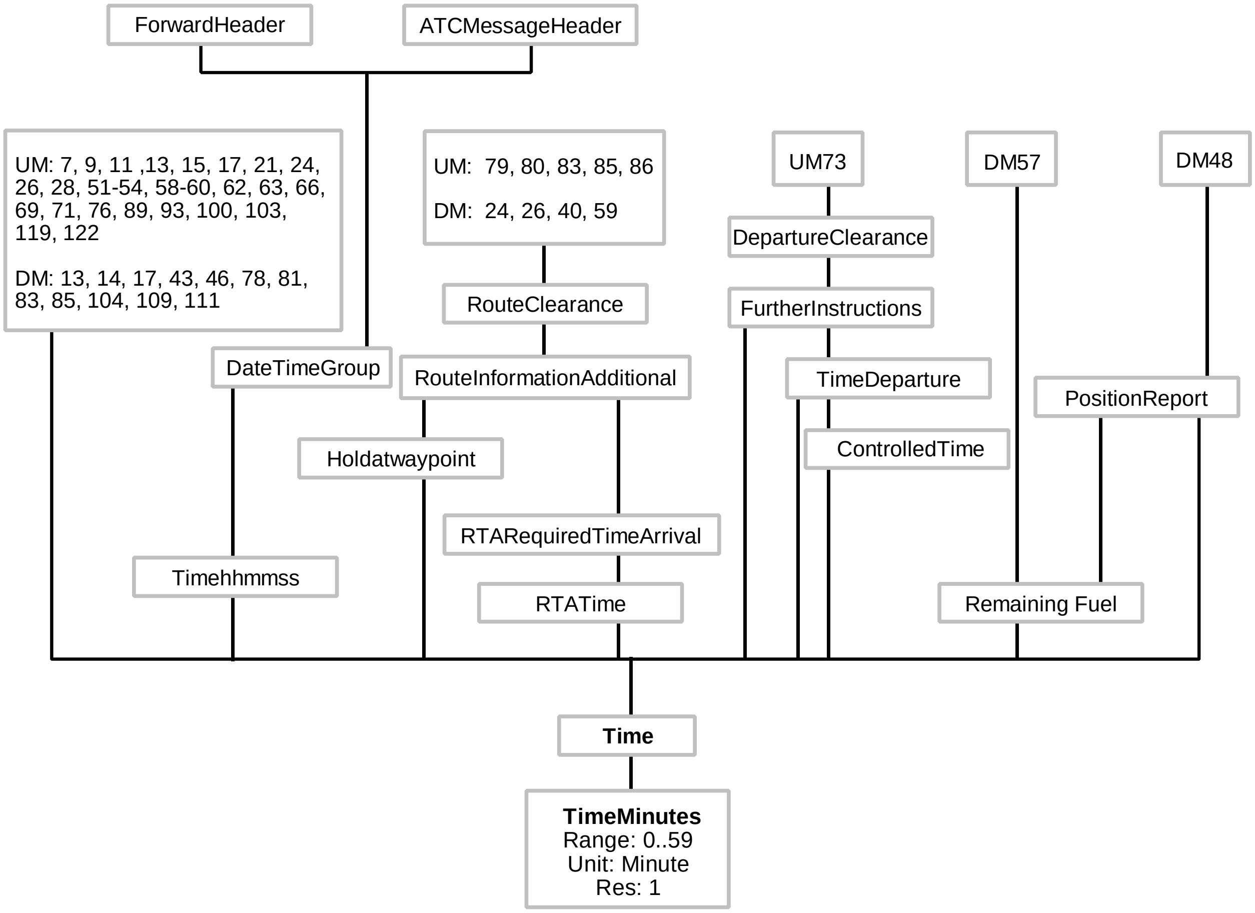


Figure 4.6-131. TimeMinutes Variable Hierarchy

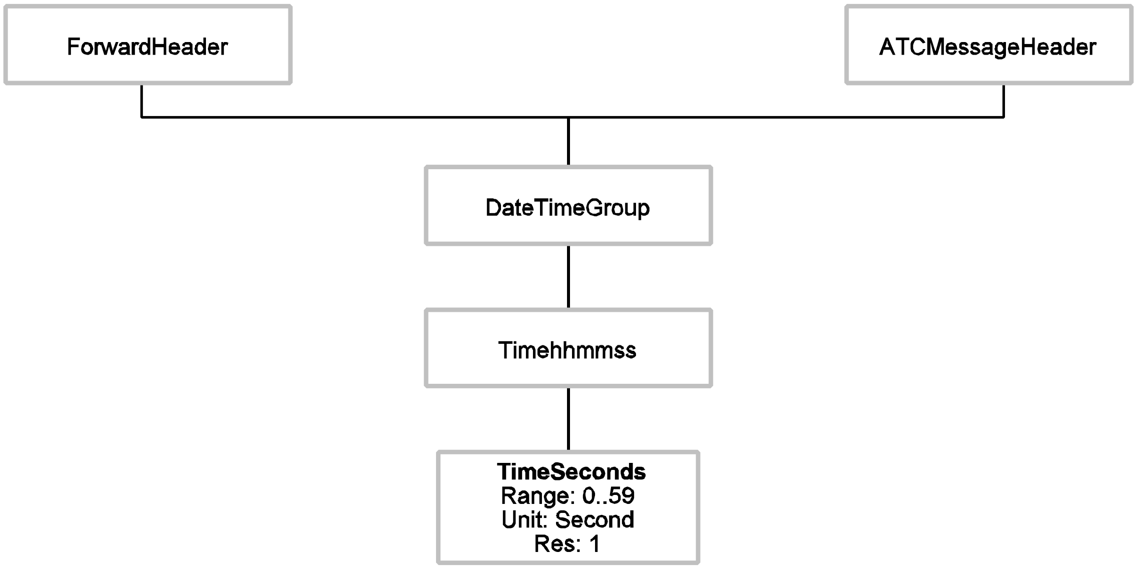


Figure 4.6-132. TimeSeconds Variable Hierarchy

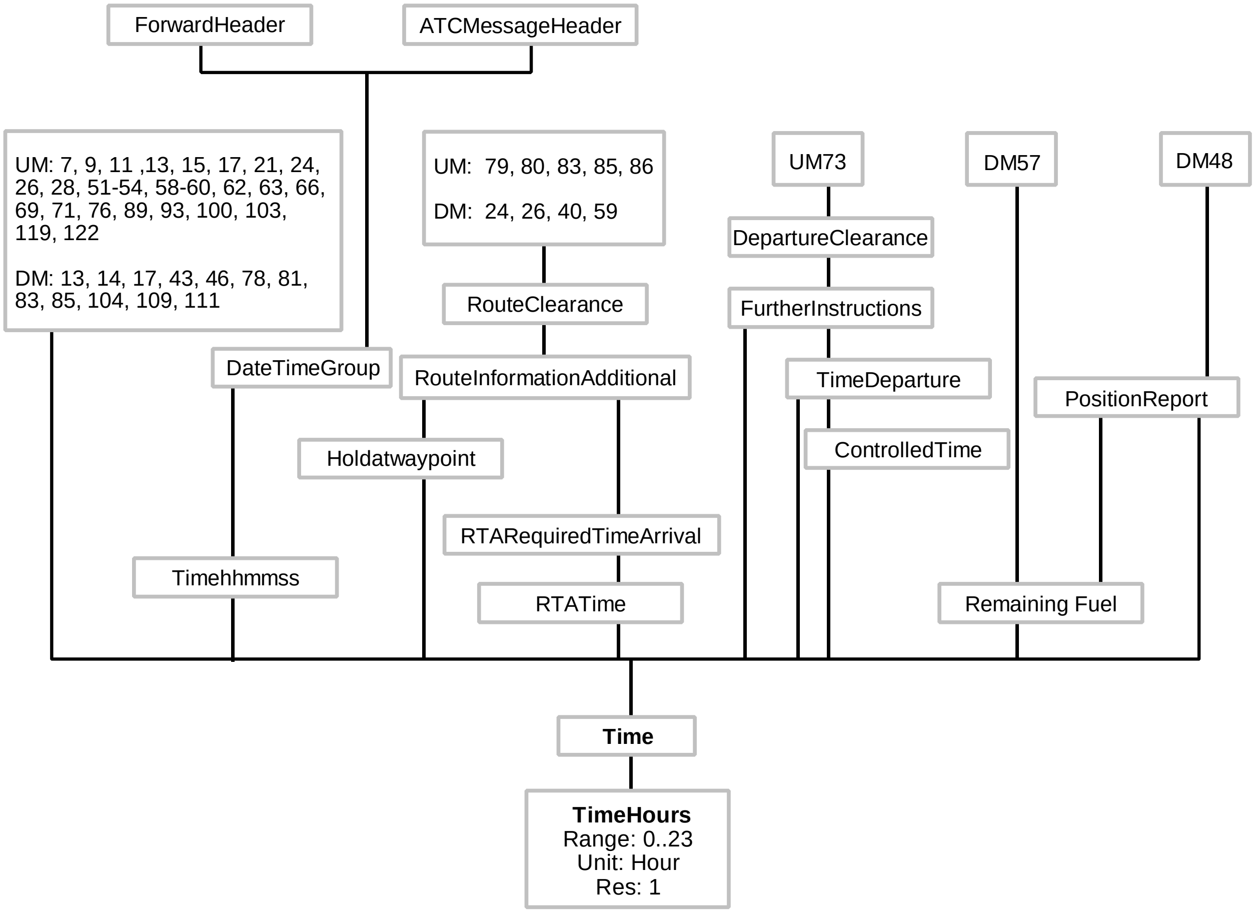


Figure 4.6-133. TimeTolerance Variable Hierarchy

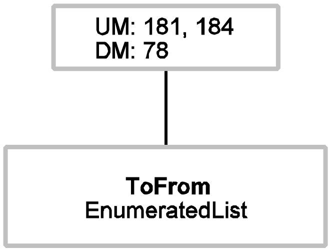


Figure 4.6-134. ToFrom Variable Hierarchy

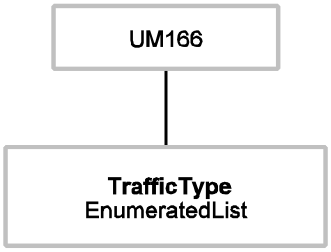


Figure 4.6-135. TrafficType Variable Hierarchy

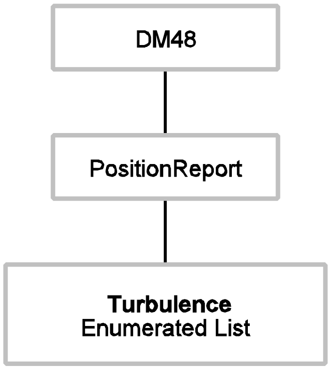


Figure 4.6-136. Turbulence Variable Hierarchy

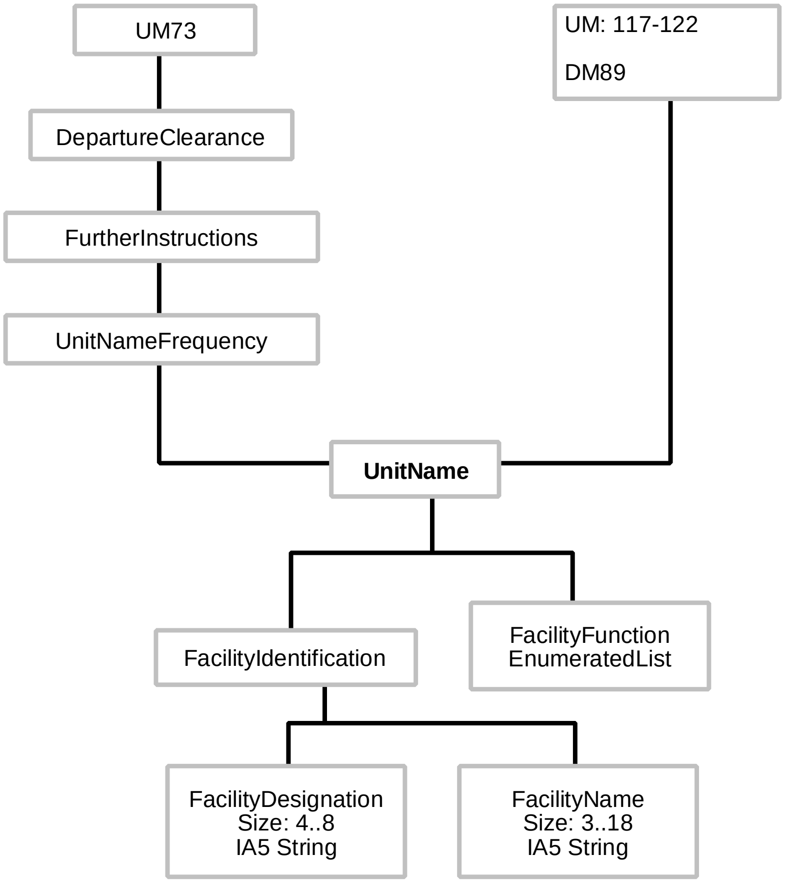


Figure 4.6-137. UnitName Variable Hierarchy

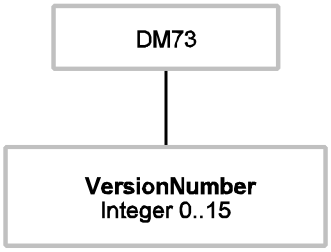


Figure 4.6-138. VersionNumber Variable Hierarchy

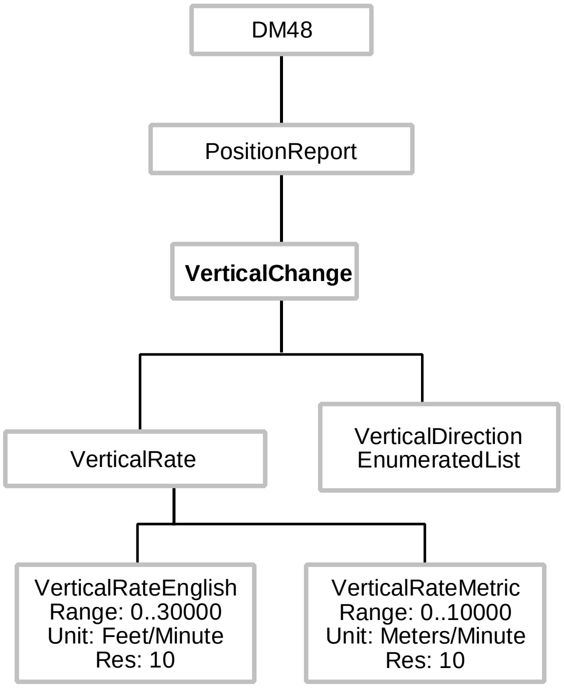

DM48
PositionReport
VerticalChange
VerticalRate
VerticalDirection EnumeratedList
VerticalRateEnglish Range: 0..30000 Unit: Feet/Minute Res: 10
VerticalRateMetric Range: 0..10000 Unit: Meters/Minute Res: 10

Figure 4.6-139. VerticalChange Variable Hierarchy

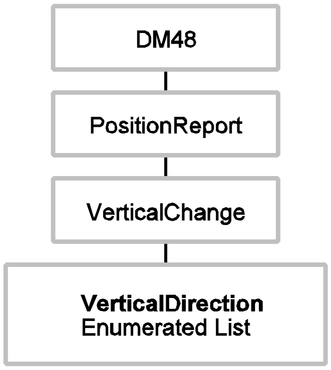


Figure 4.6-140. VerticalDirection Variable Hierarchy

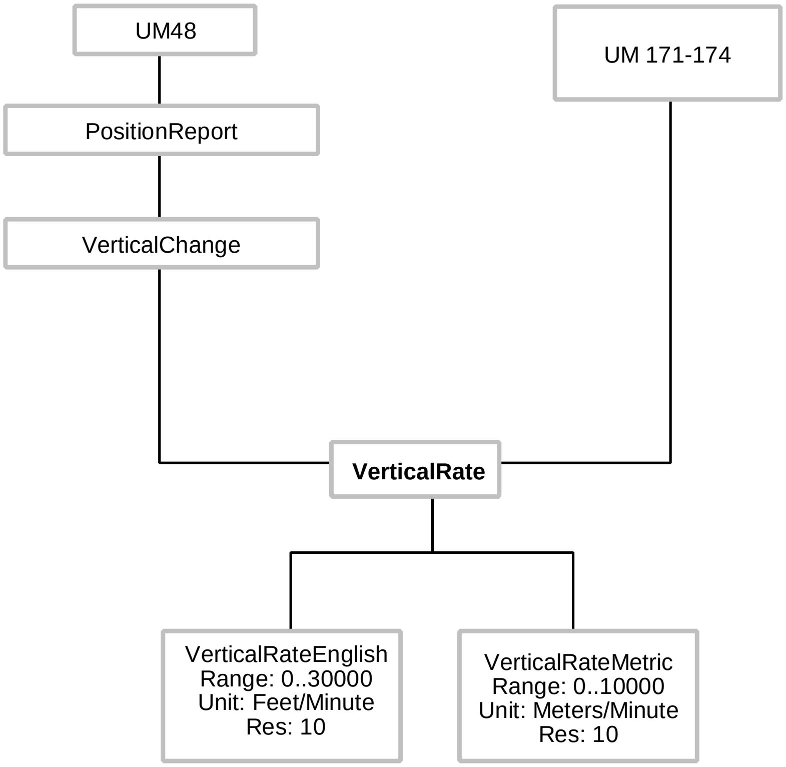


Figure 4.6-141. VerticalRate Variable Hierarchy

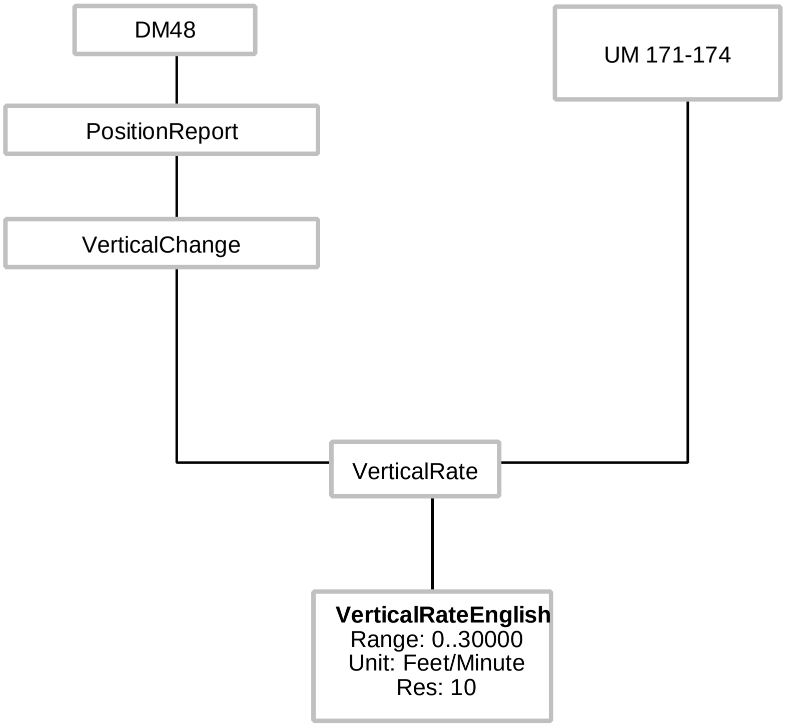


Figure 4.6-142. VerticalRateEnglish Variable Hierarchy

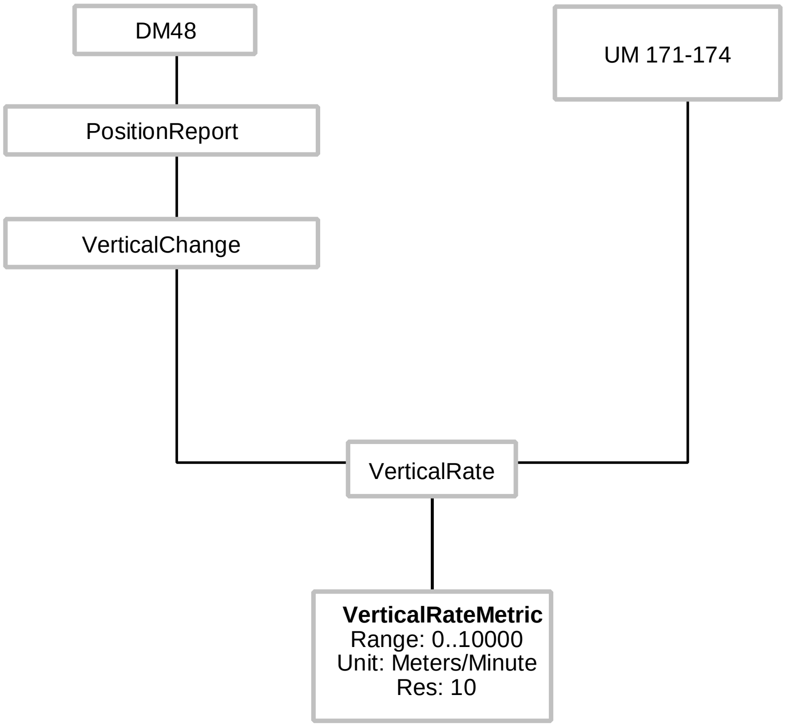


Figure 4.6-143. VerticalRateMetric Variable Hierarchy

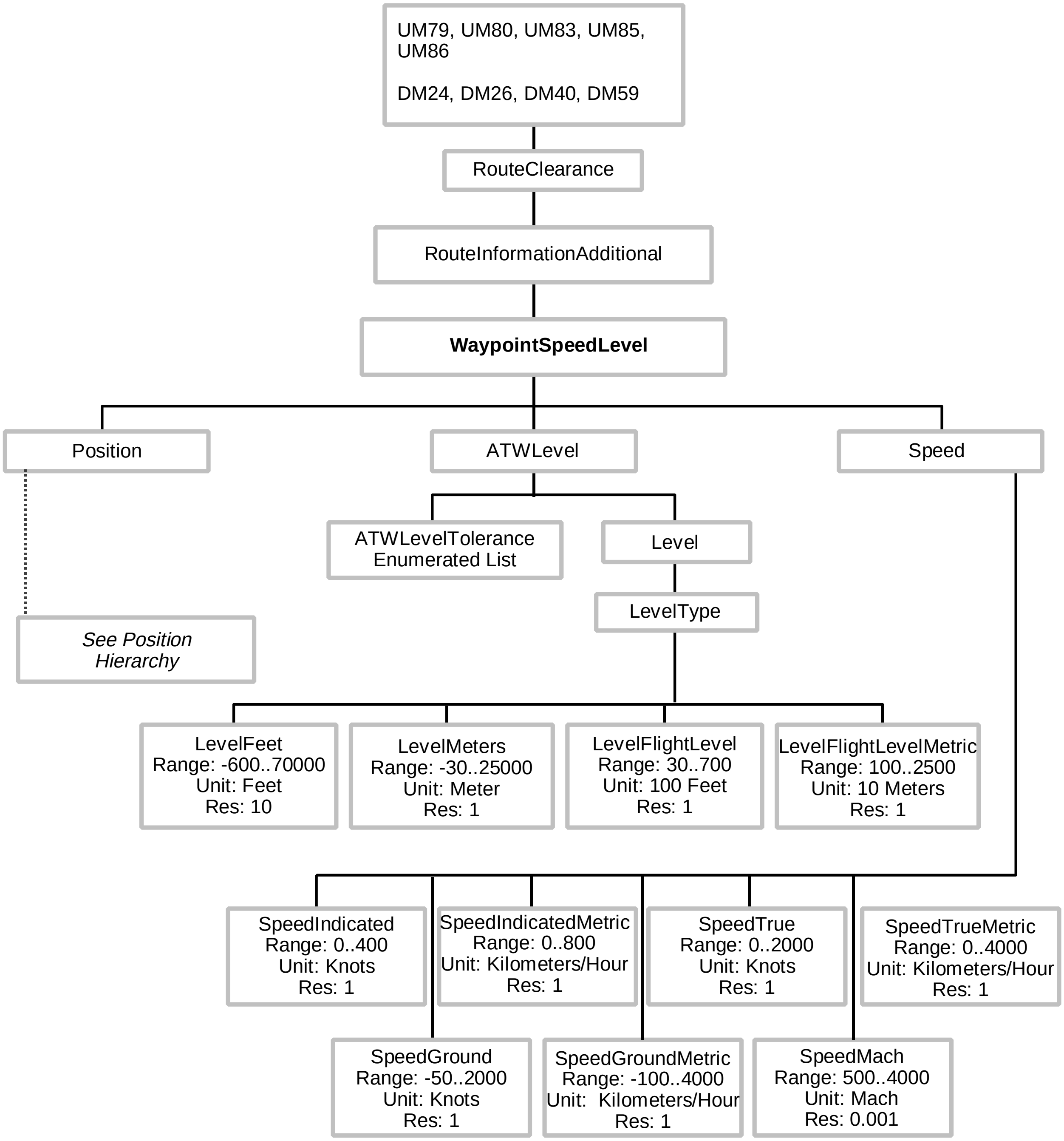

UM79, UM80, UM83, UM85, UM86
DM24, DM26, DM40, DM59
RouteClearance
RouteInformationAdditional
WaypointSpeedLevel
Position
ATWLevel
Speed
See Position Hierarchy
ATWLevelTolerance Enumerated List
Level
LevelType
LevelFeet Range: -600..70000 Unit: Feet Res: 10
LevelMeters Range: -30..25000 Unit: Meter Res: 1
LevelFlightLevel Range: 30..700 Unit: 100 Feet Res: 1
LevelFlightLevelMetric Range: 100..2500 Unit: 10 Meters Res: 1
SpeedIndicated Range: 0..400 Unit: Knots Res: 1
SpeedIndicatedMetric Range: 0..800 Unit: Kilometers/Hour Res: 1
SpeedTrue Range: 0..2000 Unit: Knots Res: 1
SpeedTrueMetric Range: 0..4000 Unit: Kilometers/Hour Res: 1
SpeedGround Range: -50..2000 Unit: Knots Res: 1
SpeedGroundMetric Range: -100..4000 Unit: Kilometers/Hour Res: 1
SpeedMach Range: 500..4000 Unit: Mach Res: 0.001

Figure 4.6-144. WaypointSpeedLevel Variable Hierarchy

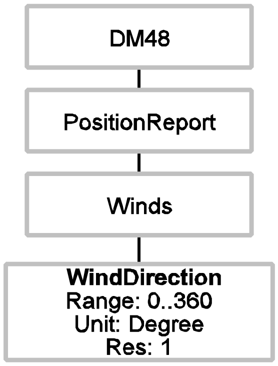


Figure 4.6-145. WindDirection Variable Hierarchy

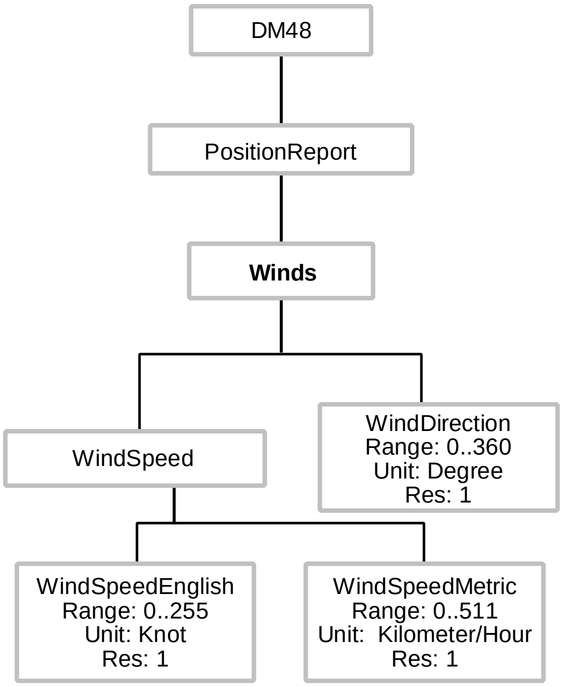


Figure 4.6-146. Winds Variable Hierarchy

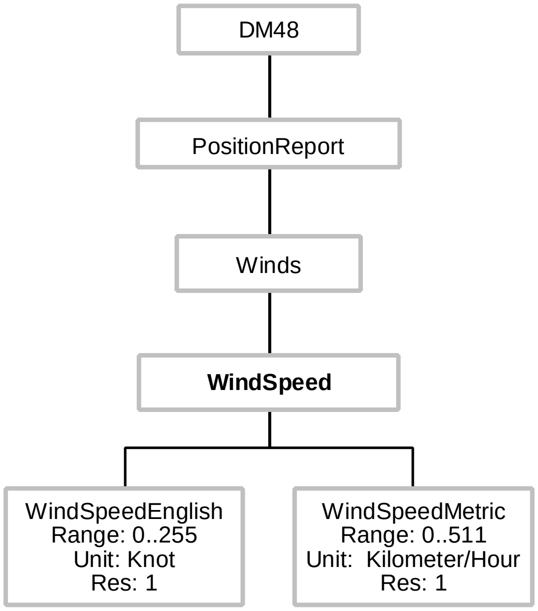


Figure 4.6-147. WindSpeed Variable Hierarchy

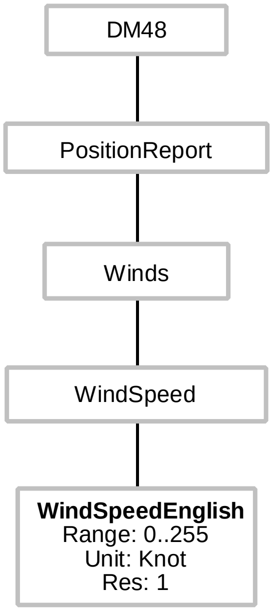


Figure 4.6-148. WindSpeedEnglish Variable Hierarchy

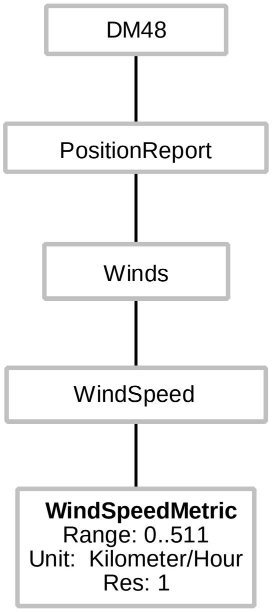


Figure 4.6-149. WindSpeedMetric Variable Hierarchy

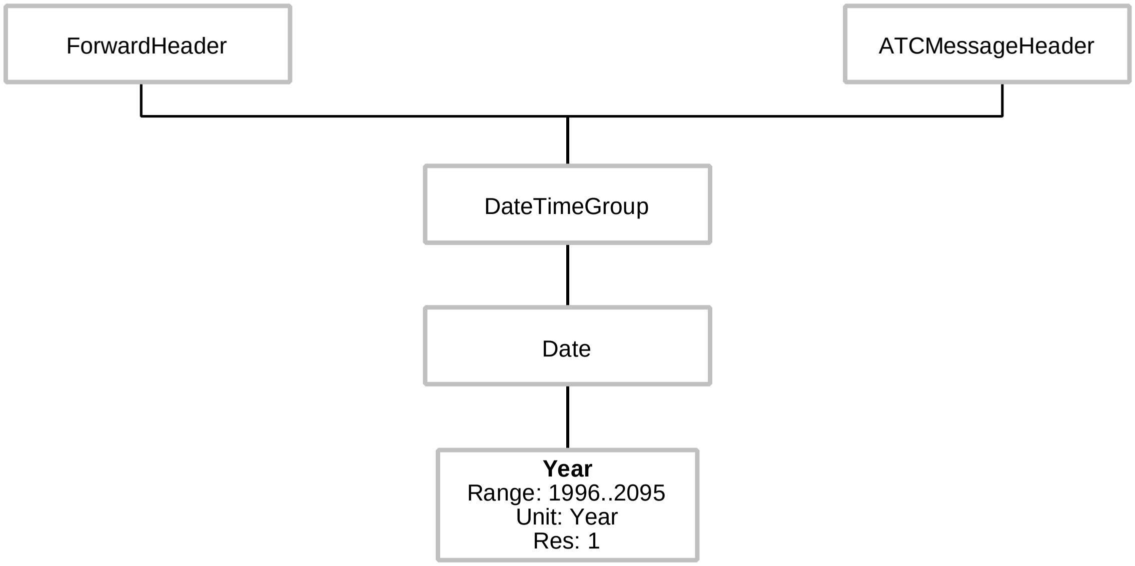


Figure 4.6-150. Year Variable Hierarchy

4.6.2 **ASN.1 Variable Definitions**

4.6.2.1 The following data are used as the CPDLC message variables, or component of the variables, and are shown here in the alphabetic order:

Aircraft Address: Unique 24-bit identifier for an aircraft.

Aircraft Flight Identification: Field 7 of the ICAO flight plan.

Airport: Four characters that specifies the ICAO four-letter identifier for the airport.

ATSRouteDesignator: Specifies the particular airway to be used within the route of the aircraft.

Altimeter: Indicates the aircraft altimeter setting in metric or English measurement units.

ATIS Code: Specifies the alphanumeric value for the current version of the automatic terminal information service (ATIS) in effect at a given location.

ATW Along Track Waypoint: Sequence of information used to compute additional way-points to an aircraft's route of flight. The following data composes the *ATW Along Track Waypoint*:

a) *Position*,

b) *ATW Distance*,

c) *Speed* (optional), and

d) *ATW Level Sequence* (optional).

ATW Distance: Used to specify the distance along a route of flight at which point to add the fix. Composed of *ATW Distance Tolerance* and *Distance*.

ATW Distance Tolerance: Indicates whether a distance can be plus or minus.

ATW Level: Contains *ATW Level Tolerance* and *Level*

ATW Level Tolerance: Indicates the vertical tolerance factor for level clearances. Used in level clearances to indicate the acceptable vertical clearance of an aircraft relative to a particular level. Indicates:

a) at,

b) at or above, or

c) at or below.

Clearance Type: Specifies a particular type of clearance. Where specified, the following clearance types are permitted:

a) approach,

b) departure,

c) further,

d) start-up,

e) pushback

f) taxi,

g) take-off,

h) landing,

i) oceanic,

j) en-route, or

k) downstream.

Code: Specifies the Mode A value (beacon code) for the aircraft. A sequence of 4 *CodeOctalDigit*.

Code Octal Digit: An octal digit used in *Code*.

Controlled Time: The time at which a flight will be released by ATC for departure.

Date: Gives the date in YYMMDD format using *Year*, *Month*, and *Day* data.

Date Time Group: Provides date and time as YYMMDD and HHMMSS.

Day: Day of the month.

Degree Increment: Specifies the number of degrees (of latitude or longitude) separating reporting points.

Degrees: Indicates the degree value in degrees magnetic or degrees true.

Departure Clearance: Sequence of data structures necessary to provide a departure clearance. The sequence of data structures that compose a departure clearance data structure are:

a) *Aircraft Identification,*

b) *Clearance Limit,*

c) *Flight Information,*

d) *Further Instructions (optional).*

Departure Minimum Interval: Specifies the minimum interval of time to depart behind the preceding aircraft.

Direction: Indicates the horizontal direction specified in terms of the current direction relative to the aircraft or in terms of the cardinal points of the compass. Values are as indicated:

a) left,

b) right

c) either side,

d) north,

e) south,

f) east,

g) west,

h) north east,

i) north west,

j) south east, or

k) south west.

Direction Degrees: A sequence of *Direction* and *Degrees*

Distance: Provides the distance in metric or English units.

Distance Offset: Specifies the offset distance from the aircraft's route in metric or English units.

Distance Offset Direction: Sequence of *Distance Offset* and *Direction* data.

Distance Offset Direction Time: Sequence of *Distance Offset* and *Direction* and *Time* data.

Error Information: Indicates the error conditions as follows:

a) unrecognized message reference number,

b) logical acknowledgment not accepted,

c) insufficient resources

d) invalid message element combination, or

e) invalid message element.

Facility: Provides a choice of whether or not to indicate a *Facility*.

Facility Designation: Specifies the ICAO four to eight letter location indicator for the facility.

Facility Designation Altimeter: Sequence of *Facility Designation* and *Altimeter*.

Facility Designation ATIS Code: Sequence of *Facility Designation* and *ATISCode*.

Facility Function: Identifies type of facility:

a) center,

b) approach,

c) tower,

d) final,

e) ground control,

f) clearance delivery,

g) departure,

h) control, or

i) radio.

Facility Identification: Provides an ICAO facility identification as either an *ICAO Facility Designation* or a *Facility Name*.

Facility Name: Specifies the ICAO facility name.

Fix: Specifies the ICAO identifier for a given fix.

Fix Name: Sequence of *Fix* and optionally a *Latitude* and *Longitude* for the fix. .

Flight Information: Information for a route of flight. Specified as:

a) *Route of Flight*, or

b) *Levels of Flight*, or

c) *Route of Flight* and *Levels of Flight*.

Free Text: Used to convey unstructured information.

Frequency: Specifies the frequency and an indicator of the RF spectrum used for the given frequency. The types of frequency that can be provided include:

a) *Frequency HF*,

b) *Frequency VHF*,

c) *Frequency UHF*, or

d) *Frequency Sat Channel*

Frequency HF: Specifies the frequency in kilohertz

Frequency Sat Channel: Specifies the appropriate address for use with a satellite voice system.

Frequency UHF: Specifies the frequency in megahertz.

Frequency VHF: Specifies the frequency as megahertz.

Further Instructions: Provides additional information in a departure clearance as follows:

a) *Code* (optional),

b) *Departure Frequency* (optional),

c) *Clearance Expiry Time* (optional),

d) *Departure Airport* (optional),

e) *Destination Airport* (optional),

f) *Departure Time* (optional),

g) *Departure Runway* (optional)

h) *Revision Number* (optional), or

i) *ATIS Code* (optional).

Hold At Waypoint: Sequence of data structures used to define the holding procedure to be used at a particular point. The *Hold At Waypoint* consists of an sequence of the following:

a) *Position*,

b) *Hold At Waypoint Speed Low* (optional),

c) *ATW Level* (optional),

d) *Hold At Waypoint Speed High* (optional),

e) *Direction* (optional),

f) *Degrees* (optional),

g) *EFC Time* (optional), and

h) *Legtype* (optional).

Hold Clearance: Provides a holding clearance to the aircraft. The *Hold Clearance* is provided using:

a) *Position*,

b) *Level*,

c) *Degrees*,

d) *Direction*, and

e) *Legtype* (optional).

Humidity: Provides humidity in percent.

Icing: Provides icing as trace, light, moderate, or severe.

Intercept Course From: The *Intercept Course From* is used to specify a fix and a bearing from that fix needed to intercept a route using *Intercept Course From Selection* and *Degrees*.

Intercept Course From Selection. Used to specify the point from which the intercept course originates and an indication of which type of fix is specified. Provided as one of the following:

a) *Published Identifier*,

b) *Latitude Longitude*,

c) *Place Bearing Place Bearing*, or

d) *Place Bearing Distance*.

Latitude: Provides latitude as *LatitudeDegrees*, or *LatitudeDegreesMinutes*, or *LatitudeDegreesMinuteSeconds*, and indicates north or south.

Latitude Degrees: Degrees of latitude.

Latitude Degrees Minutes: Specifies Latitude in whole degrees and minutes of a degree.

Latitude Degrees Minutes Seconds: Specifies Latitude in whole degrees, whole minutes of a degree and seconds of a degree.

Latitude Direction: Indicates whether north or south latitude is specified.

Latitude Longitude: Sequence of *Latitude* and *Longitude* both of which cab be optional.

Latitude Reporting Points: Indicates the latitude on which to base incremental reporting points. Provided as of *Latitude Direction* and *Latitude Degrees*.

Latitude Type: Choice of specifying latitude in Degrees, whole degrees and minutes of a degree, or whole degrees, whole minutes of a degree and seconds of a degree.

Latitude Whole Degrees: Specifies latitude in whole degrees.

Latlon Reporting Points: Provides either *Latitude Reporting Points* or *Longitude Reporting Points*.

Latlon Whole Minutes: Provides whole minutes of a degree for either Latitude or Longitude.

Leg Distance: Indicates the aircraft leg in metric or English distance units.

Leg Time: Specifies aircraft leg in terms of minutes.

Leg Type: Provides either *Leg Distance* or *Leg Time*.

Level: Specifies the Level as a single or block level.

Level Position: A sequence of *Level* and *Position*.

Level Feet: Specifies level in hundreds of feet.

Level Flight Level: Provides level as a flight level.

Level Flight LevelMetric: Provides level as a metric flight level.

Level Metric: Specifies level in meters.

Level Procedure Name: A sequence of *Level* and *Procedure Name*.

Levels of Flight: Specified as a choice of:

a) *Level*, or

b) *Procedure Name*, or

c) *Level* and *Procedure Name*.

Level Speed: A sequence of *Level* and *Speed*.

Level Time: A sequence of *Level* and *Time*.

Level Type: Specifies the Level in one of the following ways:

a) *Level* in feet or meters,

b) *Flight Level* in feet or meters.

Longitude: Provides longitude as *LongitudeDegrees*, or *LongitudeDegreesMinutes*, or *LongitudeDegreesMinuteSeconds*, and indicates east or west.

Longitude Degrees: Degrees of longitude.

Longitude Degrees Minutes: Specifies Longitude in whole degrees and minutes of a degree.

Longitude Degrees Minutes Seconds: Specifies Longitude in whole degrees, whole minutes of a degree and seconds of a degree.

Longitude Direction: Indicates whether east or west longitude is specified.

Longitude Reporting Points: Indicates the longitude on which to base incremental reporting points. Provided as *Longitude Direction* and *Longitude Degrees*.

Longitude Type: Choice of specifying longitude in Degrees, whole degrees and minutes of a degree, or whole degrees, whole minutes of a degree and seconds of a degree.

Longitude Whole Degrees: Specifies longitude in whole degrees.

Minutes Latlon: Minutes of a degrees for either latitude of longitude to hundredths of a minute.

Month: Month of the year.

Navaid: Specifies a particular navigation aid and optionally a lat/lon of the navaid.

Navaid Name: Specifies a particular navaid.

Persons On Board: Specifies the number of persons on the aircraft.

Place Bearing: Sequence of *Published Identifier* and *Degrees*.

Place Bearing Distance: Used to indicate a location based on the degrees and distance from a known point. Provided using *Published Identifier, Degrees* and *Distance* data.

Place Bearing Place Bearing: Used to define a point as the intersection formed by two bearings from two known points. Provided as two *Place Bearing*.

Position: Information used to specify a location. Position can be specified as:

a) *Fix Name*,

b) *Navaid*,

c) *Airport*,

d) *Latitude Longitude*, or

e) *Place Bearing Distance*.

Position Degrees: Sequence of *Position* and *Degrees*.

Position Distance Offset Direction: Sequence of *Position* and *Distance Offset Direction*.

Position Level: Sequence of *Position* and *Level*.

Position Level Speed: Sequence of *Position*. *Level* and *Speed*.

Position Procedure Name: Sequence of *Position* and *Procedure Name*.

Position Report: Uses the following data necessary to provide an aircraft position report as follows:

a) *Position Current*,

b) *Time At Position Current*,

c) *Level*,

d) *Fix Next* (optional),

e) *Time ETA At Fix Next* (optional),

f) *Fix Next Plus One* (optional),

g) *Time ETA destination* (optional),

h) *Remaining Fuel* (optional),

i) *Temperature* (optional),

j) *Winds* (optional),

k) *Turbulence* (optional),

l) *Icing* (optional),

m) *Speed* (optional),

n) *Speed Ground* (optional),

o) *Vertical Change* (optional),

p) *Track Angle* (optional),

q) *True Heading* (optional),

r) *Distance* (optional),

s) *Humidity* (optional),

t) *Reported Waypoint Position* (optional),

u) *Reported Waypoint Time* (optional), and

v) *Reported Waypoint Level* (optional).

Position Route Clearance: Sequence of *Position* and *Route Clearance*.

Position Speed: Sequence of *Position* and *Speed*.

PositionTime: Sequence of *Position* and *Time*.

Position Time Level: Sequence of *Position* and *Time* and *Level*.

Position Unit Name Frequency: Sequence of *Position*, *Unit Name* and *Frequency*.

Procedure: Specifies the name of the procedure.

Procedure Name: Used to uniquely identify the standard arrival, approach or departure procedure using the following:

a) *Procedure Type*,

b) *Procedure*, and

c) *Procedure Transition* (optional).

Procedure Transition: Specifies the name of the procedure transition.

Procedure Type: Specifies the type of procedure as arrival, approach, or departure.

Published Identifier: Used to indicate a *Fix Name* or a *Navaid*.

Remaining Fuel: Specifies the amount of fuel remaining on the aircraft using *Time* data.

Remaining Fuel Persons on Board: Sequence of *Remaining Fuel* and *Persons on Board*.

Reporting Points: Used to indicate reporting points along a route of flight based on a specific Latitude and/or Longitude increment expressed in degrees.

Revision Number: Specifies the revision number of the departure clearance. Used to differentiate different revisions of the departure clearance for a given aircraft flight.

Route And Levels: Sequence of *Route Information* and *Levels Of Flight*

Route Clearance: Data necessary to provide a route clearance. Provided using the following data:

a) *Departure Airport* (optional),

b) *Destination Airport* (optional),

c) *Runway Departure* (optional),

d) *Procedure Departure* (optional),

e) *Runway Arrival* (optional),

f) *Procedure Approach* (optional),

g) *Procedure Arrival* (optional),

h) *Route Information Sequence* (optional), and

i) *Route Information Additional* (optional).

Route Information: Indicate the method used to define the aircraft route of flight. The actual aircraft route of flight will probably consist of multiple *Route Information* sequences as follows:

a) *Published Identifier* (optional)

b) *Latitude Longitude* (optional),

c) *Place Bearing Place Bearing* (optional),

d) *Place Bearing Distance* (optional), and

e) *ATSRouteDesignator* (optional)

Route Information Additional: Additional data used to further specify a route clearance. Provided using the following:

a) *ATW Along Track Waypoint Sequence* (optional),

b) *Reporting Points* (optional),

c) *Intercept Course From Sequence* (optional),

d) *Hold At Waypoint Sequence* (optional),

e) *Waypoint Speed Level Sequence* (optional), and

f) *RTA Required Time Arrival* Sequence (optional).

RTA Required Time Arrival: Sequence used to associate an estimated time of arrival with a specific point along a route of flight. The *RTA Required Time Arrival* consists of:

a) *Position*,

b) *RTA Time*, and

c) *RTA Tolerance* (optional).

RTA Time: Used to specify the required time of arrival for an aircraft at a specific point.

RTA Tolerance: Specifies the possible tolerance expressed in minutes in the RTA time.

Runway: Specifies a runway using *Runway Direction* and *Runway Configuration*.

Runway Configuration: Used to specifically identify one runway in a group of parallel runways. Can be specified as left, right, or center.

Runway Direction: Specifies the direction of the runway.

Runway RVR: Sequence of *Runway* and *RVR*.

RVR: Runway Visual Range as distance in feet or meters.

Seconds Lat Lon: Seconds of minutes of latitude or longitude degrees.

Speed: Provides the aircraft speed as one of the following:

a) *Speed Indicated*,

b) *Speed True*,

c) *Speed Ground*, or

d) *Speed Mach*.

Speed Ground: Ground speed expressed in either English or metric units.

Speed Indicated: Indicated aircraft speed expressed in either English or metric units.

Speed Mach: Aircraft speed specified as a Mach value.

Speed Time: Sequence of *Speed* and *Time*.

Speed True: Aircraft true speed expressed in either English or metric units.

Speed Type: Indicates what type of speed is to be provided:

a) Indicated,

b) True,

c) Ground,

d) Mach, or

e) Not specified.

Temperature: Temperature specified in Celsius.

Time: Sequence of *Hours* and *Time Minutes*.

Time Departure: Time of departure given as an allocated time, controlled departure time, a departure clearance time, and a departure minimum interval.

Time Distance Offset Direction: Sequence of *Time* and *Distance Offset Direction*

Time Distance To From Position: Sequence of *Time, Distance, To From,* and *Position*

Time HHMMSS: Provides time as HHMMSS.

Time Hours: Time in Hours of a day.

Time Level: Sequence of *Time* and *Level*.

Time Minutes: Specifies time in minutes of an hour.

Time Position: Sequence of *Time* and *Position*.

Time Position Level: Sequence of *Time*, *Position*, and *Level*.

Time Position Level Speed: Sequence of *Time*, *Position*, *Level*, and *Speed*.

Time Seconds: Specifies time in seconds of a minute.

Time Speed: Sequence of *Time* and *Speed*.

Time To From Position: Sequence of *Time*, *To From*, and *Position*.

Time Tolerance: Provides a time tolerance as: at, at or before, or at or after.

Time Unit Name Frequency: Sequence of *Time*, *UnitName*, and *Frequency*.

To From: Specifies to or from.

To From Position: Used to indicate a “to” or “from” relative to a specified position.

Traffic Type: Indicates what type of traffic is present. Permitted types:

a) opposite direction,

b) same direction,

c) converging,

d) crossing, or

e) diverging

Turbulence: Specifies the severity of turbulence. Can be one of the following: “light”, “moderate”, or “severe”.

Unit Name: Sequence of *ICAO Facility Identification* and *ICAO Facility Function*.

Unit Name Frequency: Sequence of *ICAO Unit Name* and *Frequency*.

Version Number: Specifies version number of CPDLC.

Vertical Change: Sequence of *Vertical Direction* and *Vertical Rate*.

Vertical Direction: Specifies whether the rate of vertical change is in the upward or downward direction.

Vertical Rate: Specifies the vertical rate of change in English or metric units.

Waypoint Speed Level Used to associate levels and speeds with particular points in a route clearance. It is composed using the following:

a) *Position*,

b) *Speed* (optional), and

c) *ATW Level Sequence* (optional).

Wind Direction: Specifies the direction of the wind using *Degree Value*.

Winds: Provides wind using *Wind Direction* and *Wind Speed*.

Wind Speed: Provides wind speed in metric or English measurement units.

Year: Provides year.

4.7 Examples of Operational Scenarios

4.7.1 Introduction

4.7.1.1 This section contains a set of example scenarios of use of the CPDLC services. The purpose of this section is to demonstrate different scenarios that are possible using CPDLC. It is not meant to exhaustively cover all technical or operational possibilities. The scenarios do not attempt to depict the actual "bits over the wire" for the parameter values of the CPDLC-services and Dialogue services. For the CPDLC messages in particular, the message element is presented in a "readable" format so that the reader can get an idea of what operational information is being exchanged. The full ASN.1 construct is not shown. ASN.1 examples and PER encoding can be found in section 4.8 of this chapter.

4.7.1.2 Tables 4.7-1 provides an index into the scenarios in this section.

Table 4.7-1. Index/Summary of Sample Scenarios

Figure	Description	Ancillary Information
4.7-1	Air Initiated CPDLC, Ground Does Not Support Air Initiation of CPDLC	• any message included in start is lost when dialogue start rejected • the Air-ASE must always indicate mode of start ("dsc" or "cpdlc") • class of communication could also be reason for ground rejection
4.7-2	Ground Initiated CPDLC, Air Already Has CDA	• any message included in start is lost when dialogue start rejected • same scenario applies when CDA has not authorized an NDA
4.7-3	Ground Initiated CPDLC, NDA Not Authorized	• request for dialogue precedes NDA authorization
4.7-4	Air Initiated CPDLC, Ground Accepts	
4.7-5	Air Initiated CPDLC, Ground Accepts, Message in Start Requires Response	• messages cannot be included in dialogue acceptance • the CPDLC-message service is required to respond to any messages in CPDLC/DSC-start requests
4.7-6	Ground Initiated CPDLC, Air Accepts, Message in Start Requires Response	• messages cannot be included in dialogue acceptance • the CPDLC-message service is required to respond to any messages in CPDLC/DSC-start requests

Figure	Description	Ancillary Information
4.7-7	Crossing CPDLC Starts	• also applies to crossing NDA starts • messages in both starts processed • message identifiers lost when dialogue terminated (end or abort)
4.7-8	DSC Start, Ground Does Not Support DSC	• ground must be able to recognize DSC-start request, even if service not supported
4.7-9	DSC-CPDLC (CDA) Crossing Start	• DSC must be ended (end or abort) • messages in both starts processed • message identifiers lost when dialogue terminated (end or abort)
4.7-10	DSC-CPDLC (NDA) Crossing Starts	• both dialogues maintained
4.7-11	Typical CPDLC Message Exchanges	• message identification numbers • message reference numbers • logical acknowledgments on message by message basis
4.7-12	Typical CPDLC Message Exchanges, Logical Acknowledgment Prohibited	• message response on whole message • determining multiple message element response attribute
4.7-13	Use Of Logical Acknowledgment Prohibited, Message Sent Requiring Logical Acknowledgment	• ERROR is closure message, message pairing lost once ERROR sent
4.7-14	Message Exchanges When Received Messages Requires Response, But is Found In Error	• scenario depicts non-SARPs-compliant system
4.7.15	Message Exchanges When Received Message Does Not Require Response, But is Found In Error	• message identification numbers maintained independently by air and ground • messages with no response requirements do not permit use of reference numbers in responses
4.7-16	CPDLC-end, No Included Message, No Open Downlinks	• only CDA can invoke CPDLC-end request • open uplinks not a factor
4.7-17	CPDLC-end, Open Downlink(s)	• SARPs do not specify whether to accept or reject
4.7-18	Message in CPDLC-End Requires Responses, No Open Downlink Messages	• ground user prevent from sending any message once end is issued • request for logical acknowledgment from air is ignored
4.7-19	DSC-end, No Included Message, No Open Uplinks	• only air can invoke DSC-end request • open downlinks not a factor

Figure	Description	Ancillary Information
4.7-20	DSC-end, Open Uplink(s)	• SARPs do not specify whether to accept or reject
4.7-21	Message in DSC-End Requires Responses, No Open Uplink Messages	• air user prevent from sending any message once end is issued • request for logical acknowledgment from ground is ignored
4.7-22	CPDLC-forward Service, Receiver Does Not Support Forward Service	• receiver must recognize and reject • ground need not be initiator
4.7-23	CPDLC-forward Service, Version Numbers Not Equal	• version number checking must be done on ground-ground dialogues (CM is only air-ground)
4.7.24	CPDLC-forward Service, Service Successful	• service is “one-shot’
4.7-25	Ground Forwarding Successful Both Sides Sender/Receiver	
4.7-26	CPDLC-User Abort	• CPDLC-abort indications only given to active users • CPDLC aborts always contain a reason
4.7.27	CPDLC-Provider-Abort, ASE Aborts	• Originator is critical in distinguishing a CPDLC-user from a CPDLC-provider abort in a D-ABORT indication
4.7-28	CPDLC-Provider-Abort, Underlying Communication Service Aborts	

4.7.2 Scenario Sequence Diagrams

4.7.2.1 *Air Initiated CPDLC-start Service, Ground Rejection*

4.7.2.1.1 Figure 4.7-1 illustrates the primitives and parameters of the CPDLC-start and D-START services when a CPDLC-air-user requests a CPDLC dialogue that the receiving CPDLC-ground-user rejects since it does not support air-initiated CPDLC. In this example, the CPDLC-air-user does not provide a CPDLC message, nor does it request a particular class of communication. However, even if the CPDLC-air-user had provided a CPDLC message, it would be disregarded by a CPDLC-ground system rejecting the CPDLC-start request.

4.7.2.1.2 Upon receipt of a CPDLC start request from its user, a CPDLC-air-ASE must always create an APDU. This is required to indicate the “mode” even when the CPDLC-air-user does not supply a CPDLC message. An air initiated DSC-start request and an air initiated CPDLC-start request function identically, except for the setting of the mode. Operationally, a CPDLC-user must be able to distinguish between DSC and CPDLC dialogues. When the receiving CPDLC-ground-ASE determines that a D-START indication is from an aircraft

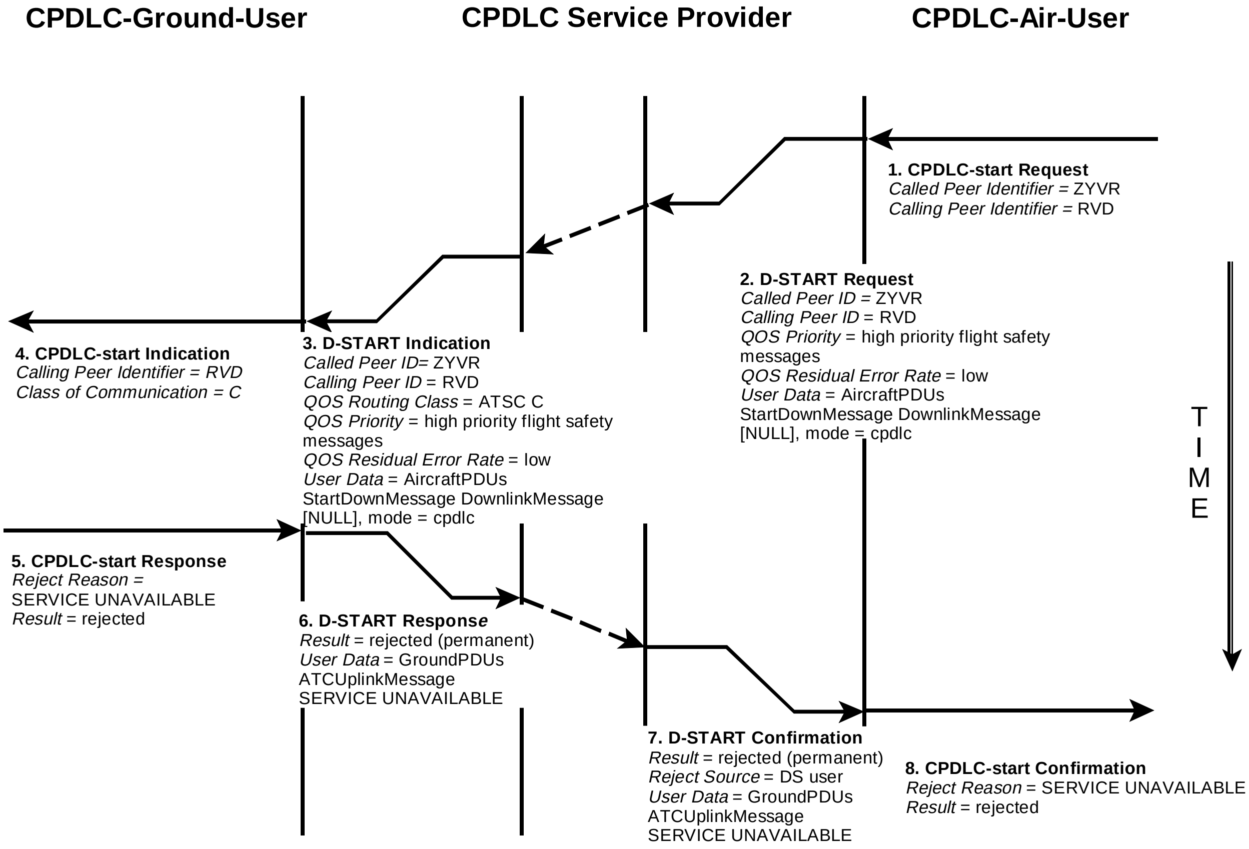

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request
Called Peer Identifier = ZYVR
Calling Peer Identifier = RVD
2. D-START Request
Called Peer ID = ZYVR
Calling Peer ID = RVD
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = cpdlc
3. D-START Indication
Called Peer ID= ZYVR
Calling Peer ID = RVD
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = cpdlc
4. CPDLC-start Indication
Calling Peer Identifier = RVD
Class of Communication = C
T I M E
5. CPDLC-start Response
Reject Reason = SERVICE UNAVAILABLE
Result = rejected
6. D-START Response
Result = rejected (permanent)
User Data = GroundPDUs ATCUplinkMessage SERVICE UNAVAILABLE
7. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = GroundPDUs ATCUplinkMessage SERVICE UNAVAILABLE
8. CPDLC-start Confirmation
Reject Reason = SERVICE UNAVAILABLE
Result = rejected

Figure 4.7-1. Air Initiated CPDLC, Ground Does Not Support Air Initiation

(and not from another ground system) it can only determine whether to invoke a CPDLC-start indication or a DSC-start indication by examining the mode. When the mode is “cpdlc” (as illustrated here) a CPDLC-start is indicated, and when the mode is “DSC” (shown in Figure 4.7-8) a DSC-start is indicated.

4.7.2.1.3 This figure also illustrates a scenario where the ground supports air initiated CPDLC, but requires a better class of communication than either selected by the CPDLC-air-user (not depicted), or selected by the underlying communication service (illustrated here). The same CPDLC-message (SERVICE UNAVAILABLE) would be supplied in the rejection.

4.7.2.2 *Ground Initiated CPDLC-start, Air Already Has CDA*

4.7.2.2.1 Figure 4.7-2 illustrates the primitives and parameters of the CPDLC-start and D-START services when a CPDLC-ground-user requests a CPDLC dialogue that the CPDLC-air-user rejects since it already has a CDA connection with another CPDLC ground system. (The existing CDA connection is not depicted in this figure.) In this example, the CPDLC-ground-user does not provide a CPDLC message, but requests a particular class of communication. However, even if the CPDLC-ground-user had provided a CPDLC message, it would be disregarded by a CPDLC-air system rejecting the CPDLC-start request.

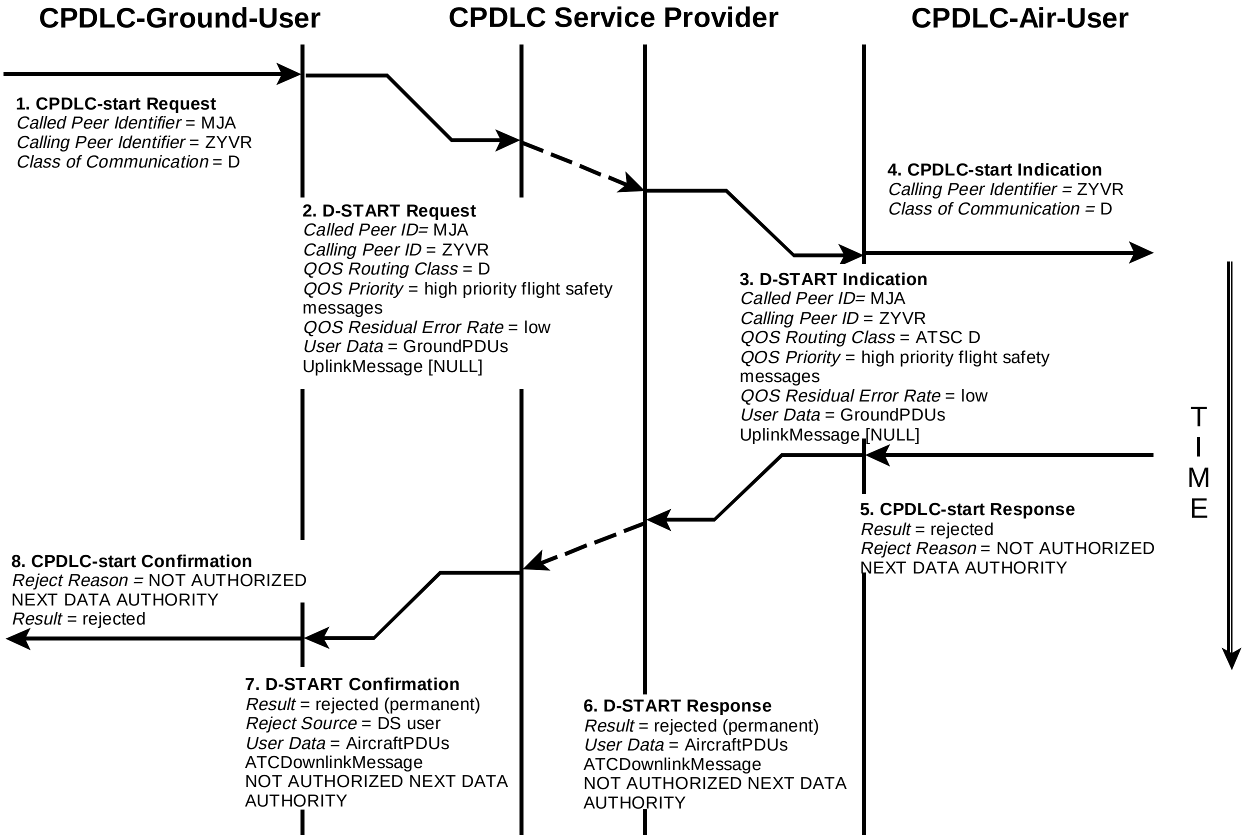

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request
Called Peer Identifier = MJA
Calling Peer Identifier = ZYVR
Class of Communication = D
2. D-START Request
Called Peer ID= MJA
Calling Peer ID = ZYVR
QOS Routing Class = D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
3. D-START Indication
Called Peer ID= MJA
Calling Peer ID = ZYVR
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
4. CPDLC-start Indication
Calling Peer Identifier = ZYVR
Class of Communication = D
5. CPDLC-start Response
Result = rejected
Reject Reason = NOT AUTHORIZED NEXT DATA AUTHORITY
6. D-START Response
Result = rejected (permanent)
User Data = AircraftPDUs ATCDownlinkMessage NOT AUTHORIZED NEXT DATA AUTHORITY
7. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = AircraftPDUs ATCDownlinkMessage NOT AUTHORIZED NEXT DATA AUTHORITY
8. CPDLC-start Confirmation
Reject Reason = NOT AUTHORIZED NEXT DATA AUTHORITY
Result = rejected
TIME

Figure 4.7-2. Ground Initiated CPDLC, Aircraft Already Has CDA

4.7.2.2.2 This scenario also illustrates 1) the CPDLC-air-user response when a CDA is in place and no ground system has been designated as a NDA by the CDA and 2). The CPDLC-air-user response when a CDA is in place, the CDA has designated a NDA, but the requesting ground system is not the ground authorized as the NDA.

4.7.2.3 Start Request From Ground System Reaches Aircraft, Before NDA Authorization Message Does

4.7.2.3.1 Figure 4.7-3 illustrates the primitives and parameters of the CPDLC-start, CPDLC-message, D-START, and D-DATA services when a CPDLC-ground-user requests a CPDLC dialogue that the CPDLC-air-user rejects. In this case the CPDLC-ground system attempts to establish a CPDLC connection prior to the message from the CDA authorizing an NDA reaching the aircraft. (The CDA connection is shown as already established.) Note that even had the NDA authorization arrived prior to the CPDLC-air-user generating a response (third row of arrows and primitives labeled 9-12 occurs before second row of arrows and primitives labeled 5-8), the CPDLC-air would still reject the NDA request. If the NDA connection was initiated by the airborne side, this situation would not occur.

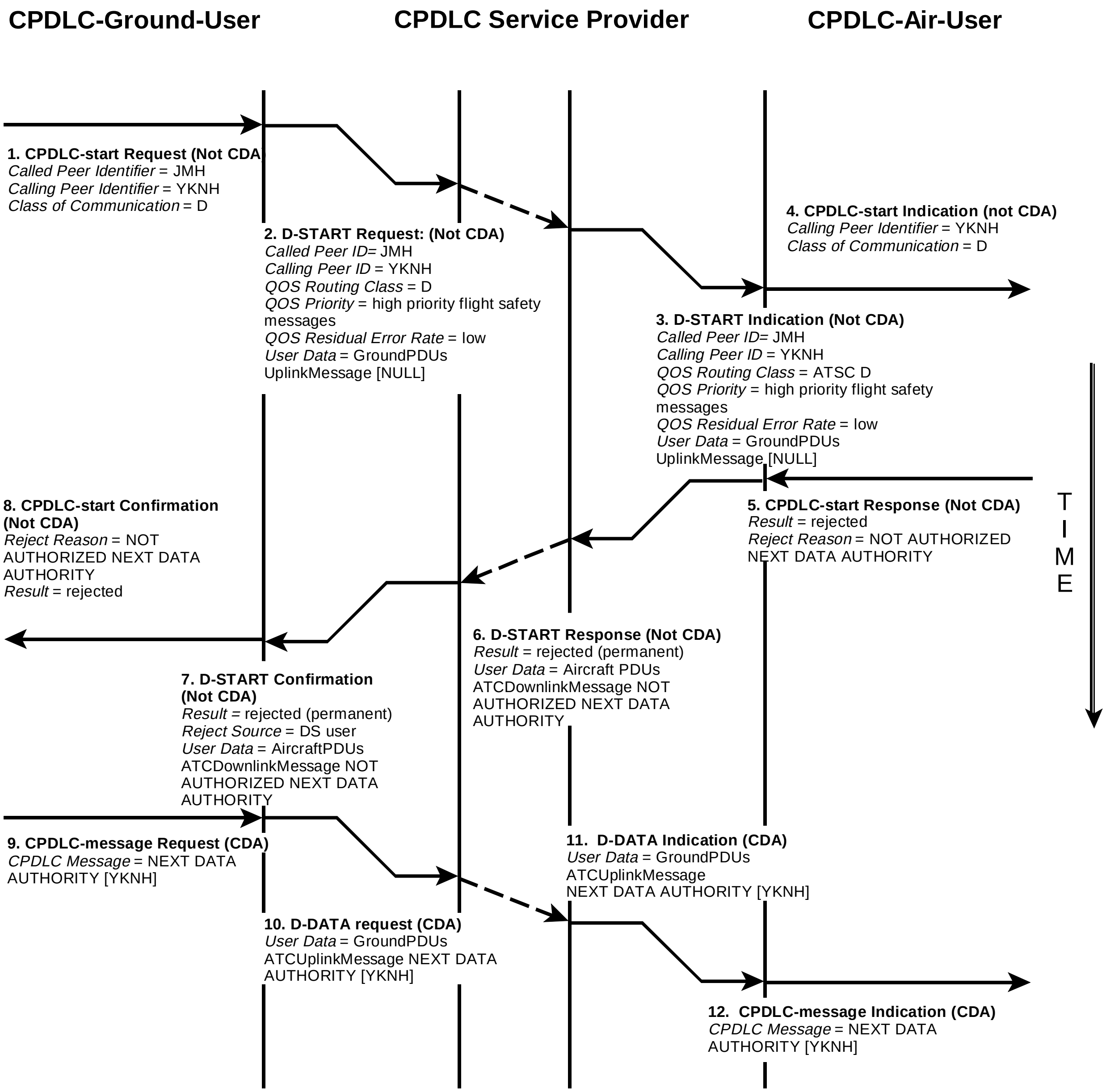

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request (Not CDA)
Called Peer Identifier = JMH
Calling Peer Identifier = YKNH
Class of Communication = D
2. D-START Request: (Not CDA)
Called Peer ID= JMH
Calling Peer ID = YKNH
QOS Routing Class = D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
4. CPDLC-start Indication (not CDA)
Calling Peer Identifier = YKNH
Class of Communication = D
3. D-START Indication (Not CDA)
Called Peer ID= JMH
Calling Peer ID = YKNH
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
8. CPDLC-start Confirmation (Not CDA)
Reject Reason = NOT AUTHORIZED NEXT DATA AUTHORITY
Result = rejected
5. CPDLC-start Response (Not CDA)
Result = rejected
Reject Reason = NOT AUTHORIZED NEXT DATA AUTHORITY
T I M E
6. D-START Response (Not CDA)
Result = rejected (permanent)
User Data = Aircraft PDUs ATCDownlinkMessage NOT AUTHORIZED NEXT DATA AUTHORITY
7. D-START Confirmation (Not CDA)
Result = rejected (permanent)
Reject Source = DS user
User Data = AircraftPDUs ATCDownlinkMessage NOT AUTHORIZED NEXT DATA AUTHORITY
9. CPDLC-message Request (CDA)
CPDLC Message = NEXT DATA AUTHORITY [YKNH]
11. D-DATA Indication (CDA)
User Data = GroundPDUs ATCUplinkMessage NEXT DATA AUTHORITY [YKNH]
10. D-DATA request (CDA)
User Data = GroundPDUs ATCUplinkMessage NEXT DATA AUTHORITY [YKNH]
12. CPDLC-message Indication (CDA)
CPDLC Message = NEXT DATA AUTHORITY [YKNH]

Figure 4.7-3. Ground Initiated CPDLC NDA Connection Reaches Aircraft Prior to NDA Authorization Message

4.7.2.3.2 This figure also illustrates the user supplying a *Class of Communication* parameter, and the underlying communication service changing the value. This is why section 2.3.3 of the CPDLC SARPs shows the indication primitive *Class of Communication* parameter as a M and not a M(=).

4.7.2.4 *Air Initiated CPDLC, Ground Accepts*

4.7.2.4.1 Figure 4.7-4 illustrates the primitives and parameters of the CPDLC-start and D-START services when a CPDLC-air-user requests a CPDLC dialogue that the CPDLC-ground-user

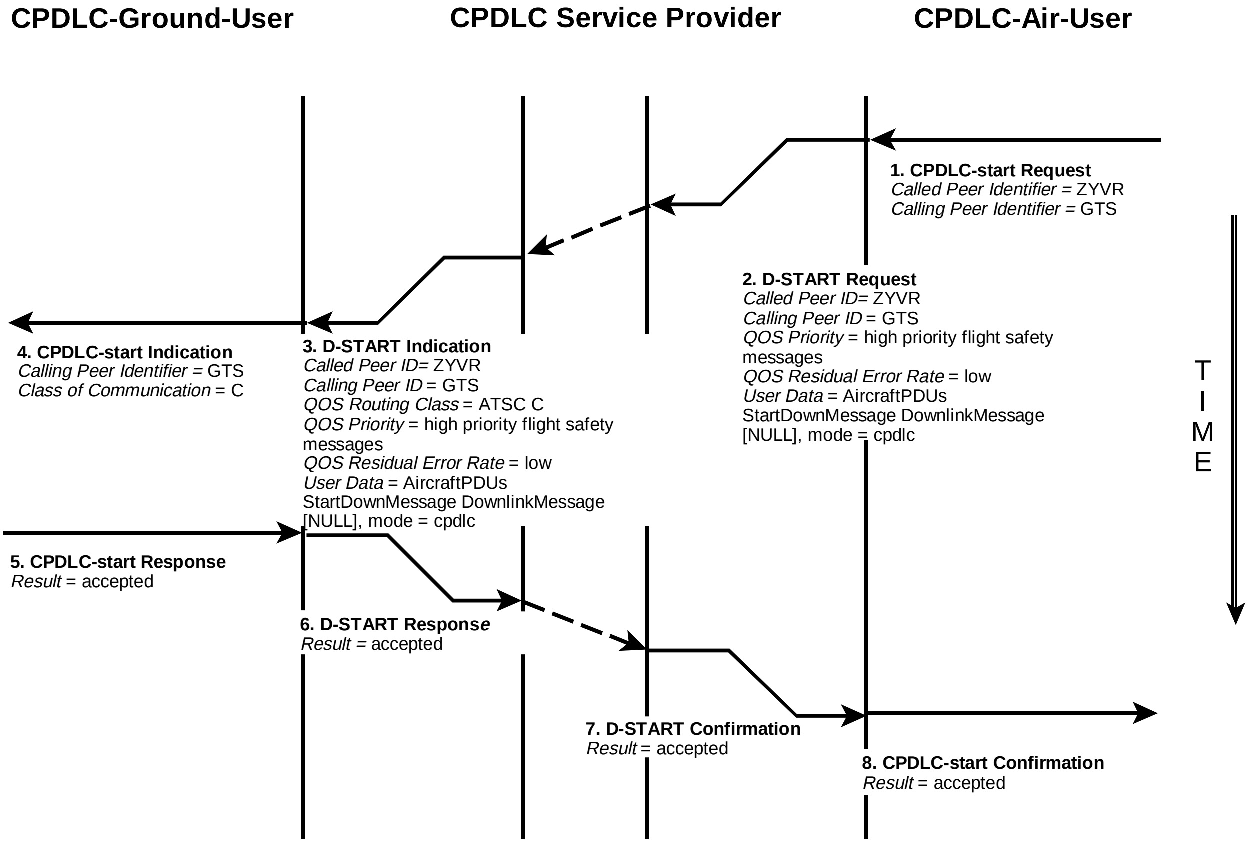

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request
Called Peer Identifier = ZYVR
Calling Peer Identifier = GTS
2. D-START Request
Called Peer ID= ZYVR
Calling Peer ID = GTS
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = cpdlc
3. D-START Indication
Called Peer ID= ZYVR
Calling Peer ID = GTS
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = cpdlc
4. CPDLC-start Indication
Calling Peer Identifier = GTS
Class of Communication = C
5. CPDLC-start Response
Result = accepted
6. D-START Response
Result = accepted
7. D-START Confirmation
Result = accepted
8. CPDLC-start Confirmation
Result = accepted
TIME

Figure 4.7-4. Air Initiated CPDLC, Ground Accepts

accepts the request. In this example, the CPDLC-air-user does not provide a CPDLC message.

4.7.2.5 ***Air Initiated CPDLC, Ground Accepts, Included Message Requires a Response***

4.7.2.5.1 Figure 4.7-5 illustrates the primitives and parameters of the CPDLC-start, CPDLC-message, D-START, and D-DATA services when a CPDLC-air-user requests a CPDLC dialogue that the CPDLC-ground-user accepts. In this example, the CPDLC-air-user provides a CPDLC message requiring both a logical acknowledgment and an operational response in the CPDLC-start request. The CPDLC SARPs do not permit a CPDLC message in a CPDLC start response when the request is accepted. Thus the CPDLC-message service is used to respond to the CPDLC message contained in the CPDLC-start request.

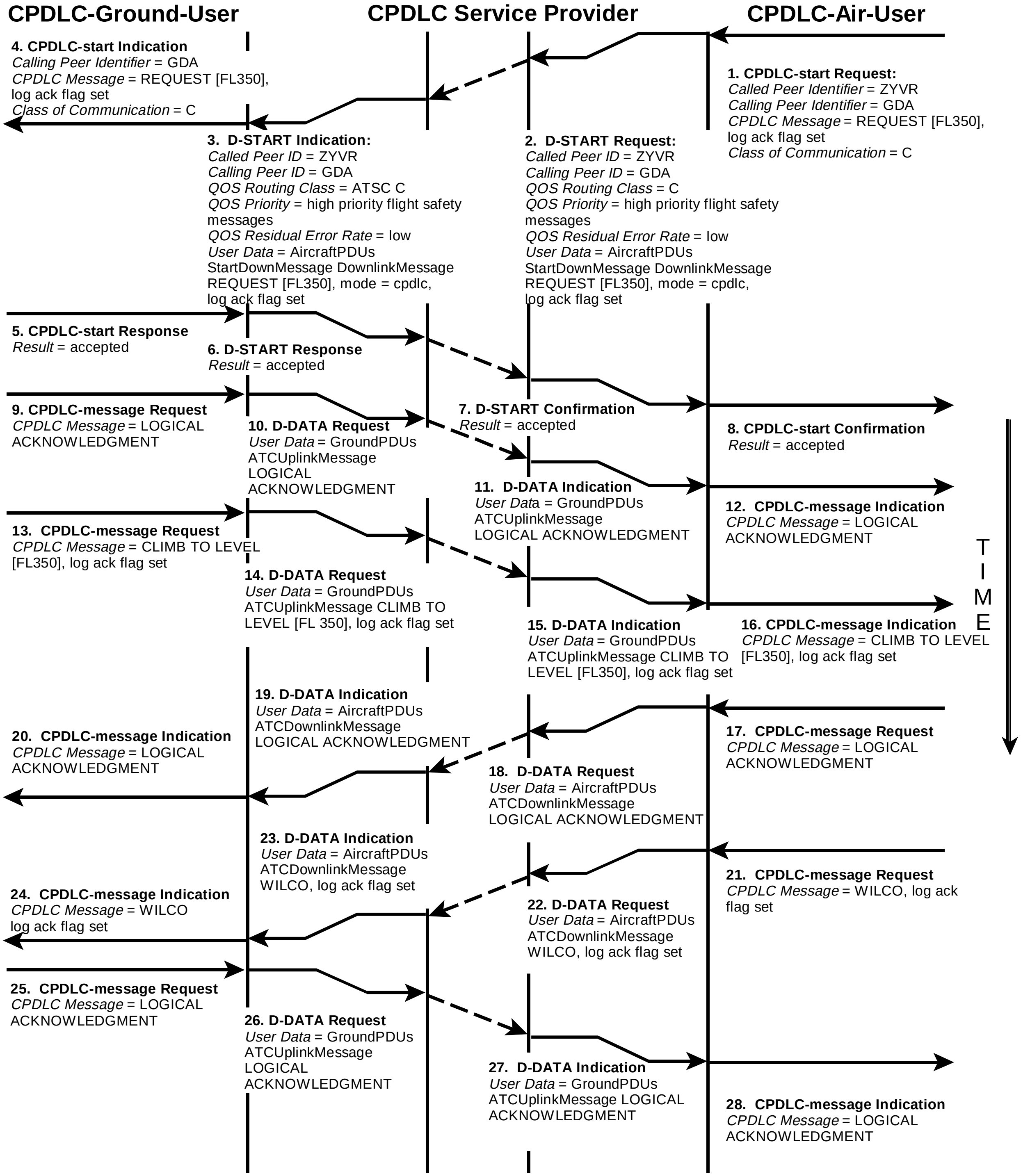

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request: Called Peer Identifier = ZYVR Calling Peer Identifier = GDA CPDLC Message = REQUEST [FL350], log ack flag set Class of Communication = C
2. D-START Request: Called Peer ID = ZYVR Calling Peer ID = GDA QOS Routing Class = C QOS Priority = high priority flight safety messages QOS Residual Error Rate = low User Data = AircraftPDUs StartDownMessage DownlinkMessage REQUEST [FL350], mode = cpdlc, log ack flag set
3. D-START Indication: Called Peer ID = ZYVR Calling Peer ID = GDA QOS Routing Class = ATSC C QOS Priority = high priority flight safety messages QOS Residual Error Rate = low User Data = AircraftPDUs StartDownMessage DownlinkMessage REQUEST [FL350], mode = cpdlc, log ack flag set
4. CPDLC-start Indication Calling Peer Identifier = GDA CPDLC Message = REQUEST [FL350], log ack flag set Class of Communication = C
5. CPDLC-start Response Result = accepted
6. D-START Response Result = accepted
7. D-START Confirmation Result = accepted
8. CPDLC-start Confirmation Result = accepted
9. CPDLC-message Request CPDLC Message = LOGICAL ACKNOWLEDGMENT
10. D-DATA Request User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT
11. D-DATA Indication User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT
12. CPDLC-message Indication CPDLC Message = LOGICAL ACKNOWLEDGMENT
13. CPDLC-message Request CPDLC Message = CLIMB TO LEVEL [FL350], log ack flag set
14. D-DATA Request User Data = GroundPDUs ATCUplinkMessage CLIMB TO LEVEL [FL 350], log ack flag set
15. D-DATA Indication User Data = GroundPDUs ATCUplinkMessage CLIMB TO LEVEL [FL350], log ack flag set
16. CPDLC-message Indication CPDLC Message = CLIMB TO LEVEL [FL350], log ack flag set
T I M E
17. CPDLC-message Request CPDLC Message = LOGICAL ACKNOWLEDGMENT
18. D-DATA Request User Data = AircraftPDUs ATCDownlinkMessage LOGICAL ACKNOWLEDGMENT
19. D-DATA Indication User Data = AircraftPDUs ATCDownlinkMessage LOGICAL ACKNOWLEDGMENT
20. CPDLC-message Indication CPDLC Message = LOGICAL ACKNOWLEDGMENT
21. CPDLC-message Request CPDLC Message = WILCO, log ack flag set
22. D-DATA Request User Data = AircraftPDUs ATCDownlinkMessage WILCO, log ack flag set
23. D-DATA Indication User Data = AircraftPDUs ATCDownlinkMessage WILCO, log ack flag set
24. CPDLC-message Indication CPDLC Message = WILCO log ack flag set
25. CPDLC-message Request CPDLC Message = LOGICAL ACKNOWLEDGMENT
26. D-DATA Request User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT
27. D-DATA Indication User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT
28. CPDLC-message Indication CPDLC Message = LOGICAL ACKNOWLEDGMENT

Figure 4.7-5. Air Initiated CPDLC, Ground Accepts, Message in Start Requires Response

4.7.2.6 *Ground Initiated CPDLC, Air Accepts, Included Message Requires a Response*

4.7.2.6.1 Figure 4.7-6 illustrates the primitives and parameters of the CPDLC-start, CPDLC-message, D-START, and D-DATA services when a CPDLC-ground-user requests a CPDLC dialogue that the CPDLC-air-user accepts. In this example, the CPDLC-ground-user provides a CPDLC message requiring both a logical acknowledgment and an operational response in the CPDLC-start request. The CPDLC SARPs do not permit a CPDLC message in a CPDLC start response when the request is accepted. Thus the CPDLC-message service is used to respond to the CPDLC message contained in the CPDLC-start request.

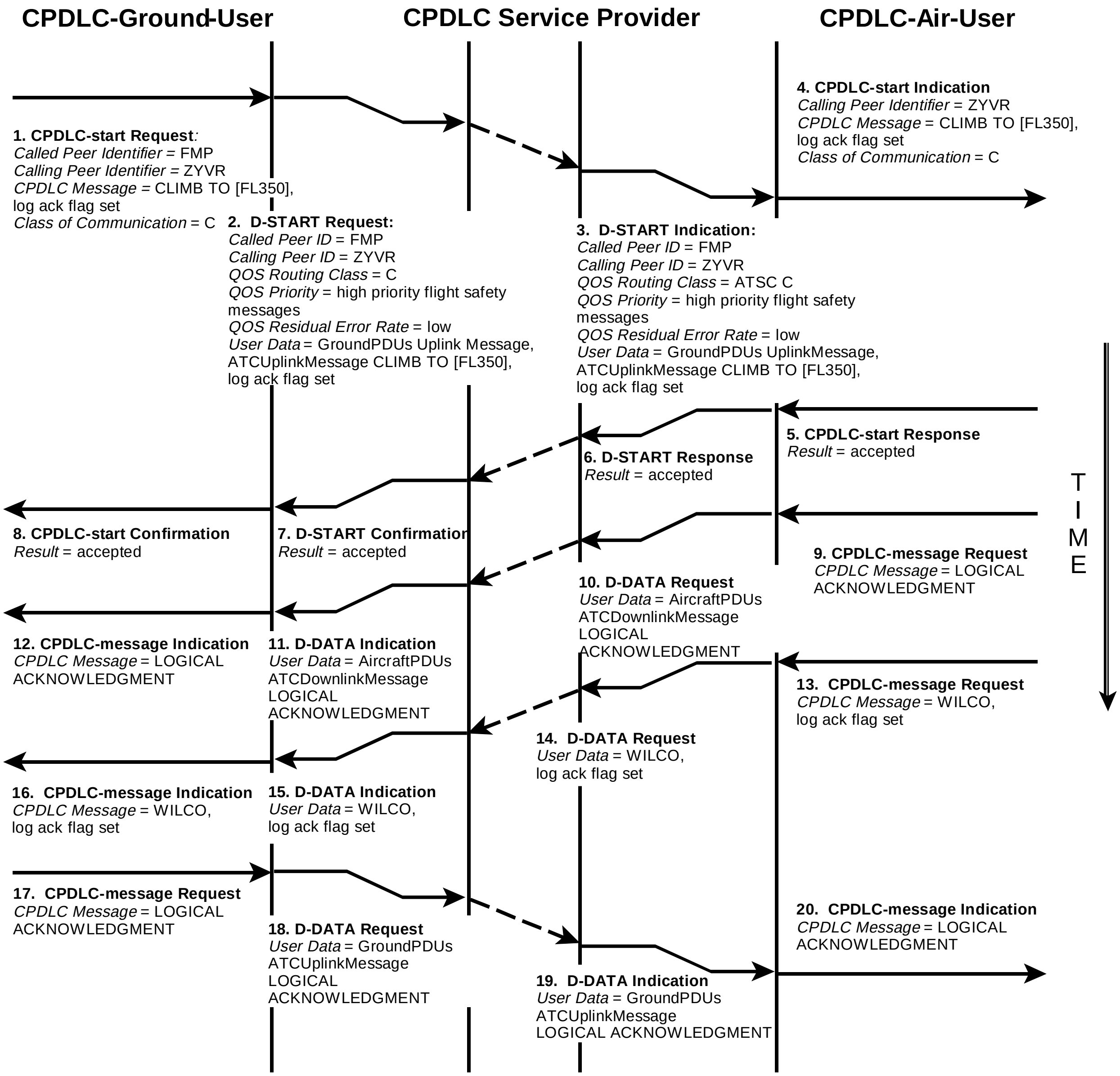


Figure 4.7-6. Ground Initiated CPDLC, Air Accepts, Message in Start Requires Response

4.7.2.7 Crossing CPDLC Starts

4.7.2.7.1 Figure 4.7-7 illustrates the primitives and parameters of the CPDLC-start, CPDLC-user-abort, D-START, and D-ABORT services when a CPDLC-ground-user requests a CPDLC dialogue that "crosses" with a CPDLC-air-user also requesting a CPDLC dialogue. In this case both connections are accepted, and the CPDLC-air-user must abort the first connection. This scenario applies to both CDA-CDA crossing starts and NDA-NDA crossing starts. Note that the response/confirmation do not necessarily cross, but could cross and are illustrated as crossing in the example.

4.7.2.7.2 This example does not show any CPDLC messages included in the start requests, but when such messages are included, they are processed (each dialogue is accepted). Any message on the subsequently aborted dialogue (in this case the ground initiated or "B" dialogue), can be responded to operationally, but there will be no associated response identifiers. Identifiers are "lost" when the dialogue is aborted.

4.7.2.7.3 This scenario shows an "A" and a "B" dialogue. They are separate dialogues and numbered independently. There is no time ordering between them.

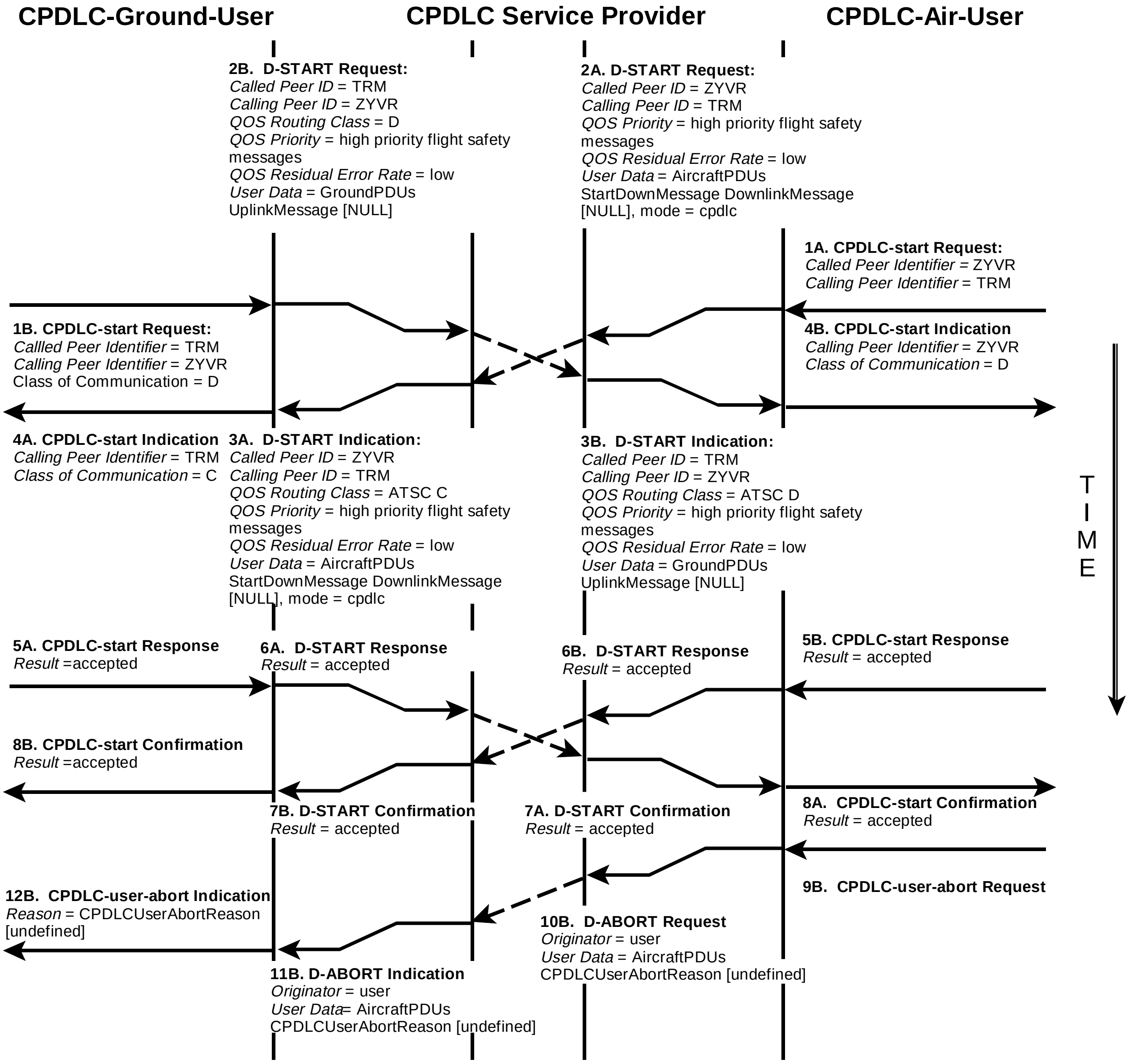

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
2B. D-START Request:
Called Peer ID = TRM
Calling Peer ID = ZYVR
QOS Routing Class = D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
2A. D-START Request:
Called Peer ID = ZYVR
Calling Peer ID = TRM
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = cpdlc
1A. CPDLC-start Request:
Called Peer Identifier = ZYVR
Calling Peer Identifier = TRM
1B. CPDLC-start Request:
Callled Peer Identifier = TRM
Calling Peer Identifier = ZYVR
Class of Communication = D
4B. CPDLC-start Indication
Calling Peer Identifier = ZYVR
Class of Communication = D
4A. CPDLC-start Indication
Calling Peer Identifier = TRM
Class of Communication = C
3A. D-START Indication:
Called Peer ID = ZYVR
Calling Peer ID = TRM
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = cpdlc
3B. D-START Indication:
Called Peer ID = TRM
Calling Peer ID = ZYVR
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
T I M E
5A. CPDLC-start Response
Result =accepted
6A. D-START Response
Result = accepted
6B. D-START Response
Result = accepted
5B. CPDLC-start Response
Result = accepted
8B. CPDLC-start Confirmation
Result =accepted
7B. D-START Confirmation
Result = accepted
7A. D-START Confirmation
Result = accepted
8A. CPDLC-start Confirmation
Result = accepted
12B. CPDLC-user-abort Indication
Reason = CPDLCUserAbortReason [undefined]
9B. CPDLC-user-abort Request
10B. D-ABORT Request
Originator = user
User Data = AircraftPDUs CPDLCUserAbortReason [undefined]
11B. D-ABORT Indication
Originator = user
User Data= AircraftPDUs CPDLCUserAbortReason [undefined]

Figure 4.7-7. Ground Initiated CPDLC Crosses Air Initiated CPDLC Two Dialogues Result, First Must be Aborted by Air

4.7.2.8 ***DSC-start Service, DSC Not Supported By Ground***

4.7.2.8.1 Figure 4.7-8 illustrates the primitives and parameters of the CPDLC-start and D-START services when a CPDLC-air-user requests a DSC dialogue that the CPDLC-ground-user rejects since it does not support DSC. In this example, the CPDLC-air-user does not provide a CPDLC message, nor request a particular class of communication. However, even if the CPDLC-air-user had provided a CPDLC message, it would be disregarded by a CPDLC-ground system rejecting the DSC-start request. Even if a CPDLC-ground system does not support the DSC capability, it must at a minimum, recognize the DSC-start request, and reject the request.

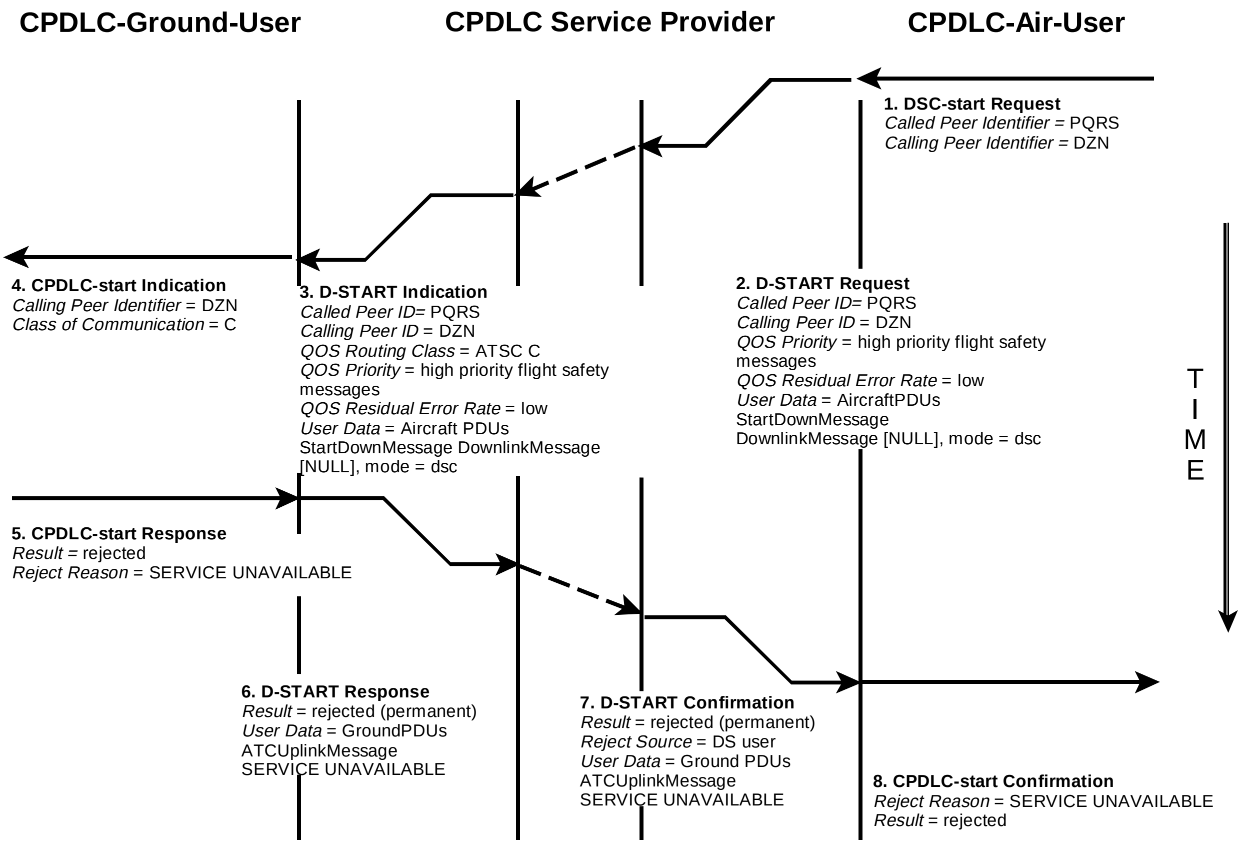

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. DSC-start Request
Called Peer Identifier = PQRS
Calling Peer Identifier = DZN
2. D-START Request
Called Peer ID= PQRS
Calling Peer ID = DZN
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = dsc
3. D-START Indication
Called Peer ID= PQRS
Calling Peer ID = DZN
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = Aircraft PDUs StartDownMessage DownlinkMessage [NULL], mode = dsc
4. CPDLC-start Indication
Calling Peer Identifier = DZN
Class of Communication = C
T I M E
5. CPDLC-start Response
Result = rejected
Reject Reason = SERVICE UNAVAILABLE
6. D-START Response
Result = rejected (permanent)
User Data = GroundPDUs ATCUplinkMessage SERVICE UNAVAILABLE
7. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = Ground PDUs ATCUplinkMessage SERVICE UNAVAILABLE
8. CPDLC-start Confirmation
Reject Reason = SERVICE UNAVAILABLE
Result = rejected

Figure 4.7-8. DSC-start Service, Ground Does Not Support DSC

4.7.2.9 *DSC-CPDLC CDA Crossing Starts, DSC Terminated*

4.7.2.9.1 Figure 4.7-9 illustrates the primitives and parameters of the CPDLC-start, DSC-start, CPDLC-user-abort, D-START, and D-ABORT services when a CPDLC-ground-user requests a CPDLC dialogue that “crosses” with a CPDLC-air-user requesting a DSC dialogue. In this scenario the aircraft has no existing CPDLC connection, so the CPDLC-start results in a CDA connection. In this case both connections are accepted, and the CPDLC-air-user must end the DSC connection. In this example the CPDLC-air-user aborts the DSC connection. The CPDLC-air-user could initiate a DSC-end as well (not depicted). The SARPs simply require that the DSC connection be terminated when a DDA becomes a CDA. Note that the response/confirmation do not necessarily cross, but could cross and are illustrated as crossing in the example.

4.7.2.9.2 This example does not show any CPDLC messages included in the start requests, but when such messages are included, they are processed (each dialogue is accepted). Any message on the subsequently aborted dialogue (the DSC is shown aborted), can be responded to operationally (in this case over the CPDLC link), but there will be no associated response identifiers. The identifiers are “lost” when the dialogue is aborted. If a message were included in the DSC start, it would probably be better to conclude the operational message exchange, and then have the CPDLC-air issue a DSC-end or an abort as illustrated.

4.7.2.9.3 This scenario shows an “A” and a “B” dialogue. They are separate dialogues and numbered independently. There is no time ordering between them.

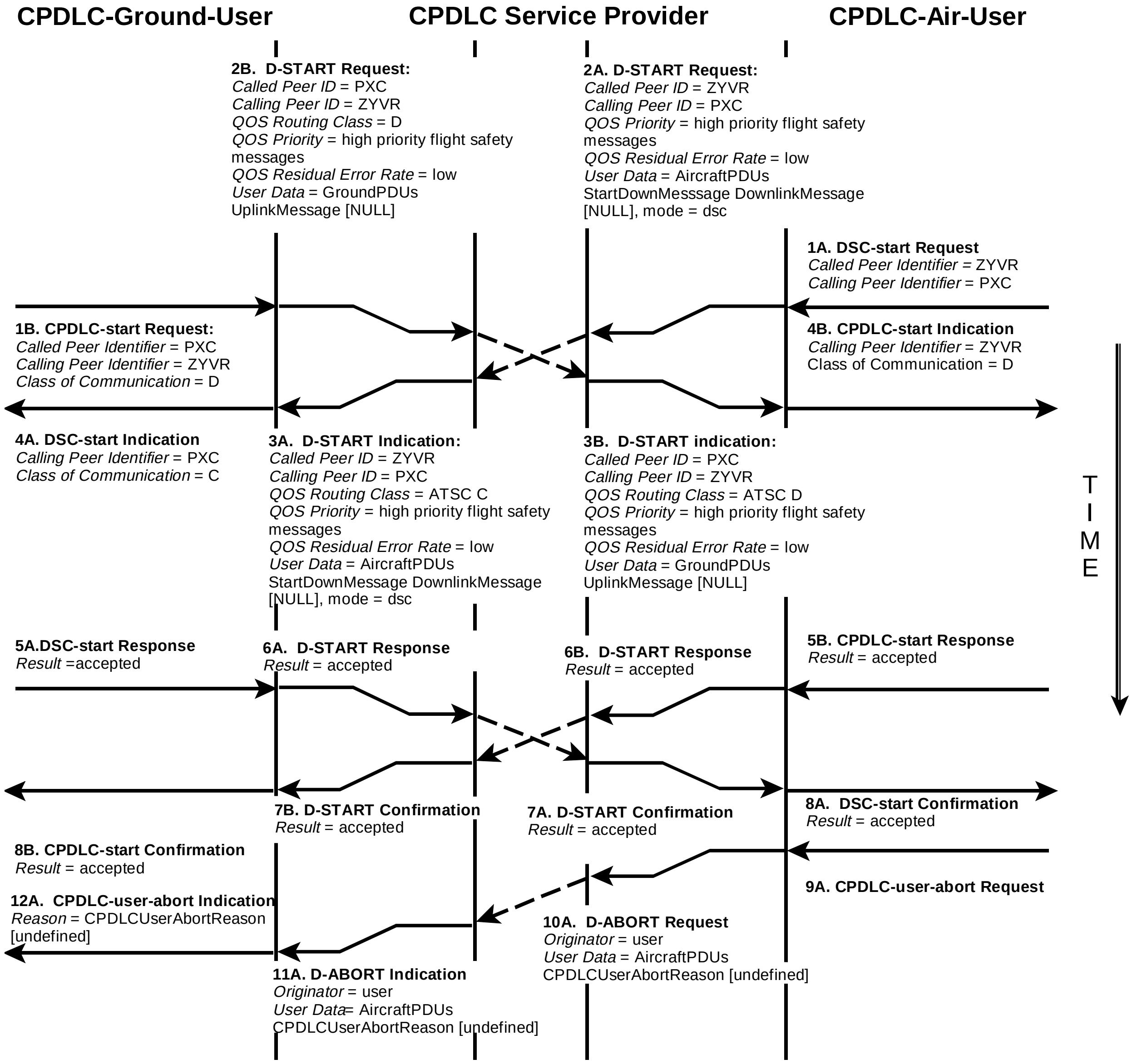

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
2B. D-START Request:
Called Peer ID = PXC
Calling Peer ID = ZYVR
QOS Routing Class = D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
2A. D-START Request:
Called Peer ID = ZYVR
Calling Peer ID = PXC
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMesssage DownlinkMessage [NULL], mode = dsc
1A. DSC-start Request
Called Peer Identifier = ZYVR
Calling Peer Identifier = PXC
1B. CPDLC-start Request:
Called Peer Identifier = PXC
Calling Peer Identifier = ZYVR
Class of Communication = D
4B. CPDLC-start Indication
Calling Peer Identifier = ZYVR
Class of Communication = D
4A. DSC-start Indication
Calling Peer Identifier = PXC
Class of Communication = C
3A. D-START Indication:
Called Peer ID = ZYVR
Calling Peer ID = PXC
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = dsc
3B. D-START indication:
Called Peer ID = PXC
Calling Peer ID = ZYVR
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
T I M E
5A.DSC-start Response
Result =accepted
6A. D-START Response
Result = accepted
6B. D-START Response
Result = accepted
5B. CPDLC-start Response
Result = accepted
7B. D-START Confirmation
Result = accepted
7A. D-START Confirmation
Result = accepted
8A. DSC-start Confirmation
Result = accepted
8B. CPDLC-start Confirmation
Result = accepted
9A. CPDLC-user-abort Request
12A. CPDLC-user-abort Indication
Reason = CPDLCUserAbortReason [undefined]
10A. D-ABORT Request
Originator = user
User Data = AircraftPDUs CPDLCUserAbortReason [undefined]
11A. D-ABORT Indication
Originator = user
User Data= AircraftPDUs CPDLCUserAbortReason [undefined]

Figure 4.7-9. Ground Initiated CPDLC Crossing Air Initiated DSC Two Dialogues Result, DSC Must be Terminated by Air

4.7.2.10 ***DSC-CPDLC NDA Crossing Starts, Both Maintained***

4.7.2.10.1 Figure 4.7-10 illustrates the primitives and parameters of the CPDLC-start, DSC-start, and D-START services when a CPDLC-ground-user requests a CPDLC dialogue that "crosses" with a CPDLC-air-user requesting a DSC dialogue. In this scenario the aircraft has an existing CDA connection, and the CPDLC-start is from an authorized NDA and thus results in a NDA connection. In this example, both connections are accepted. Both connections are operationally valid, and permitted by the SARPs, so both are maintained.

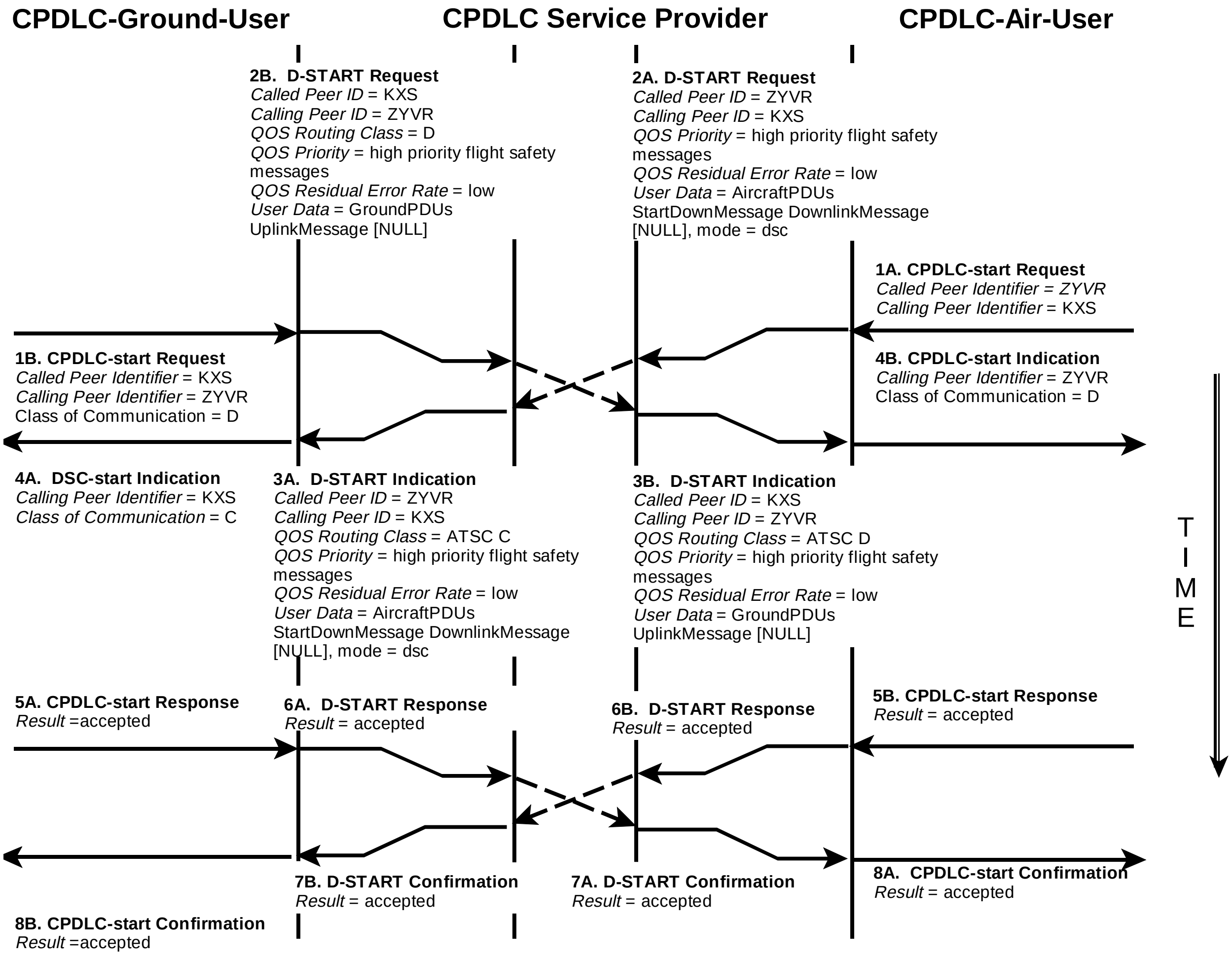

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
2B. D-START Request
Called Peer ID = KXS
Calling Peer ID = ZYVR
QOS Routing Class = D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
2A. D-START Request
Called Peer ID = ZYVR
Calling Peer ID = KXS
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = dsc
1A. CPDLC-start Request
Called Peer Identifier = ZYVR
Calling Peer Identifier = KXS
1B. CPDLC-start Request
Called Peer Identifier = KXS
Calling Peer Identifier = ZYVR
Class of Communication = D
4B. CPDLC-start Indication
Calling Peer Identifier = ZYVR
Class of Communication = D
4A. DSC-start Indication
Calling Peer Identifier = KXS
Class of Communication = C
3A. D-START Indication
Called Peer ID = ZYVR
Calling Peer ID = KXS
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = AircraftPDUs StartDownMessage DownlinkMessage [NULL], mode = dsc
3B. D-START Indication
Called Peer ID = KXS
Calling Peer ID = ZYVR
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage [NULL]
T
I
M
E
5A. CPDLC-start Response
Result =accepted
6A. D-START Response
Result = accepted
6B. D-START Response
Result = accepted
5B. CPDLC-start Response
Result = accepted
7B. D-START Confirmation
Result = accepted
7A. D-START Confirmation
Result = accepted
8A. CPDLC-start Confirmation
Result = accepted
8B. CPDLC-start Confirmation
Result =accepted

Figure 4.7-10. Ground Initiated CPDLC Crossing Air Initiated DSC Two Dialogues Result: NDA and DSC, Both Maintained

4.7.2.11 ***Typical CPDLC Message Exchanges, Dialogue Already Established***

4.7.2.11.1 Figure 4.7-11 illustrates the primitives and parameters of the CPDLC-message and D-DATA services when a CPDLC-ground-user sends a CPDLC message requiring a logical acknowledgment and an operational response. The airborne operational response also requires a logical acknowledgment. Since the CPDLC-message service is unconfirmed, all CPDLC-user CPDLC message exchanges use the CPDLC-message request/indication. An operational "response" can map to a "request" primitive. In this scenario the dialogue is already established and the CPDLC-start service is not depicted

4.7.2.11.2 This scenario also illustrates how message identification numbers and message reference numbers would work. A message identification number is assigned to every CPDLC message. The use of message reference numbers are determined by message response requirements. When required/permitted, the value of a message reference number is determined by its associated message identification number. Message identification and reference numbers (if required) apply to all scenarios depicted in this section that contain a CPDLC message but are only illustrated in selected scenarios.

4.7.2.11.3 Logical acknowledgment requirements are determined on a message by message basis. This scenario illustrates an operational environment where logical acknowledgments are required for all operational messages. The LOGICAL ACKNOWLEDGMENT message in this scenario does not itself require a logical acknowledgment. The SARPs place no requirements at all on what CPDLC messages can/cannot require a logical acknowledgment. Thus technically there is no prohibition from a LOGICAL ACKNOWLEDGMENT message requiring a logical acknowledgment, but this is not considered operationally sensible (and thus not depicted).

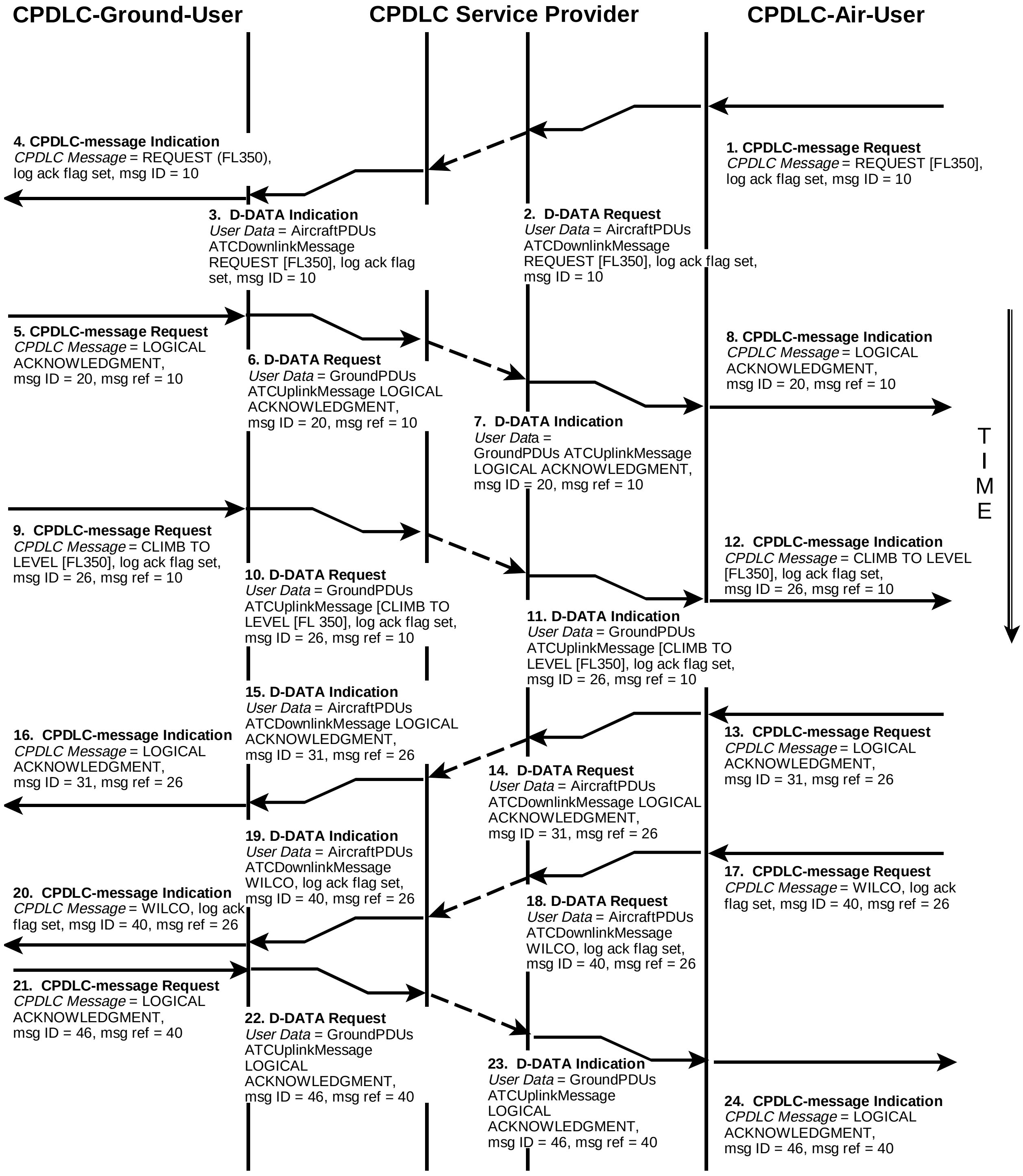

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-message Request
CPDLC Message = REQUEST [FL350], log ack flag set, msg ID = 10
2. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage REQUEST [FL350], log ack flag set, msg ID = 10
3. D-DATA Indication
User Data = AircraftPDUs ATCDownlinkMessage REQUEST [FL350], log ack flag set, msg ID = 10
4. CPDLC-message Indication
CPDLC Message = REQUEST (FL350), log ack flag set, msg ID = 10
5. CPDLC-message Request
CPDLC Message = LOGICAL ACKNOWLEDGMENT, msg ID = 20, msg ref = 10
6. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT, msg ID = 20, msg ref = 10
7. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT, msg ID = 20, msg ref = 10
8. CPDLC-message Indication
CPDLC Message = LOGICAL ACKNOWLEDGMENT, msg ID = 20, msg ref = 10
TIME
9. CPDLC-message Request
CPDLC Message = CLIMB TO LEVEL [FL350], log ack flag set, msg ID = 26, msg ref = 10
10. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage [CLIMB TO LEVEL [FL 350], log ack flag set, msg ID = 26, msg ref = 10
11. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage [CLIMB TO LEVEL [FL350], log ack flag set, msg ID = 26, msg ref = 10
12. CPDLC-message Indication
CPDLC Message = CLIMB TO LEVEL [FL350], log ack flag set, msg ID = 26, msg ref = 10
13. CPDLC-message Request
CPDLC Message = LOGICAL ACKNOWLEDGMENT, msg ID = 31, msg ref = 26
14. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage LOGICAL ACKNOWLEDGMENT, msg ID = 31, msg ref = 26
15. D-DATA Indication
User Data = AircraftPDUs ATCDownlinkMessage LOGICAL ACKNOWLEDGMENT, msg ID = 31, msg ref = 26
16. CPDLC-message Indication
CPDLC Message = LOGICAL ACKNOWLEDGMENT, msg ID = 31, msg ref = 26
17. CPDLC-message Request
CPDLC Message = WILCO, log ack flag set, msg ID = 40, msg ref = 26
18. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage WILCO, log ack flag set, msg ID = 40, msg ref = 26
19. D-DATA Indication
User Data = AircraftPDUs ATCDownlinkMessage WILCO, log ack flag set, msg ID = 40, msg ref = 26
20. CPDLC-message Indication
CPDLC Message = WILCO, log ack flag set, msg ID = 40, msg ref = 26
21. CPDLC-message Request
CPDLC Message = LOGICAL ACKNOWLEDGMENT, msg ID = 46, msg ref = 40
22. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT, msg ID = 46, msg ref = 40
23. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT, msg ID = 46, msg ref = 40
24. CPDLC-message Indication
CPDLC Message = LOGICAL ACKNOWLEDGMENT, msg ID = 46, msg ref = 40

Figure 4.7-11. CPDLC Message Exchanges, Logical Acknowledgments Required

4.7.2.12 *Typical CPDLC Message Exchanges, When Logical Acknowledgments Are Prohibited*

4.7.2.12.1 Figure 4.7-12 illustrates the primitives and parameters of the CPDLC-start, CPDLC-message, D-START, and D-DATA services when a CPDLC-ground-user requests a CPDLC dialogue and includes a CPDLC message indicating that logical acknowledgments are prohibited (UM233) and a clearance element requiring an operational response (UM20). (The CPDLC message consists of two message elements.) The CPDLC-air-user accepts the dialogue request. The CPDLC SARPs do not permit a CPDLC message in a CPDLC start response when the request is accepted. Thus the CPDLC-message service is used to respond to the CPDLC message contained in the CPDLC-start request.

4.7.2.12.2 The message prohibiting logical acknowledgments may be sent at any time a dialogue is being established or is in place. The message is not required in the start request. Once the rohibition message has been sent, logical acknowledgments are prohibited for the duration of the dialogue. A CPDLC-ground-user could also send this message once a DSC dialogue is established

4.7.2.12.3 In this example, the CPDLC message sent by the CPDLC-ground-user is composed of a message element with an “N” response attribute (UM233) and a message element with a “W/U” response attribute (UM20). However, a given CPDLC message can only be responded to as a whole message, (a response is not permitted on an element by element basis). The CPDLC SARPs section 2.3.7 Tables 2.3.7-3 and 2.3.7-4 provide the requirements for determining both the CPDLC message response attribute, by using the *Precedence* column, and provides the valid responses for each CPDLC message response attribute. In this example, the entire message has a “W/U” response attribute.

4.7.2.12.4 Note also that the “log ack” flag is not shown at all in this example. (See Figure 4.7-11 for an example where it is shown.) The CPDLC message structure ASN.1 definition defaults to not requiring a logical acknowledgment, and the flag must be explicitly set on a message by message basis when logical acknowledgment is required.

4.7.2.12.5 Operationally, only a ground system can specify that logical acknowledgment is not allowed, so there is no corresponding downlink message element for UM233.

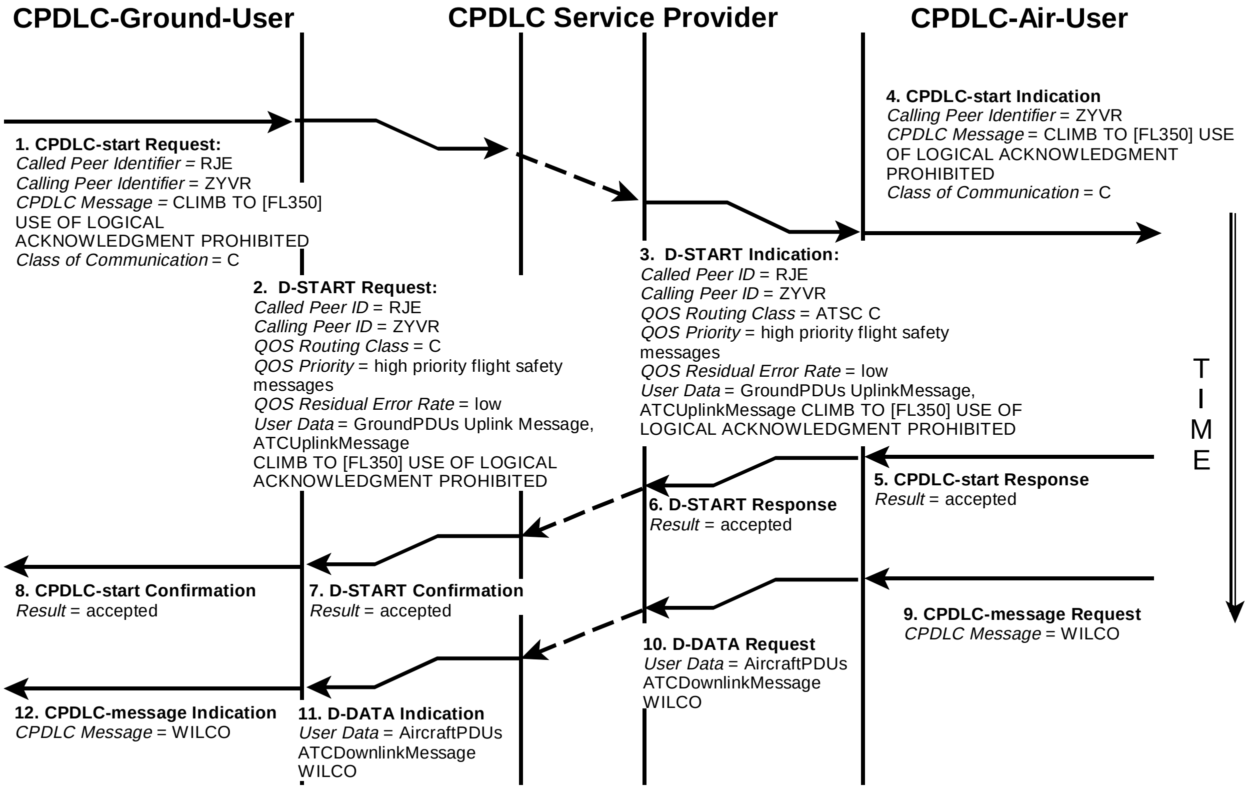

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request:
Called Peer Identifier = RJE
Calling Peer Identifier = ZYVR
CPDLC Message = CLIMB TO [FL350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED
Class of Communication = C
2. D-START Request:
Called Peer ID = RJE
Calling Peer ID = ZYVR
QOS Routing Class = C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs Uplink Message, ATCUplinkMessage CLIMB TO [FL350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED
3. D-START Indication:
Called Peer ID = RJE
Calling Peer ID = ZYVR
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage, ATCUplinkMessage CLIMB TO [FL350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED
4. CPDLC-start Indication
Calling Peer Identifier = ZYVR
CPDLC Message = CLIMB TO [FL350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED
Class of Communication = C
5. CPDLC-start Response
Result = accepted
6. D-START Response
Result = accepted
7. D-START Confirmation
Result = accepted
8. CPDLC-start Confirmation
Result = accepted
9. CPDLC-message Request
CPDLC Message = WILCO
10. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage WILCO
11. D-DATA Indication
User Data = AircraftPDUs ATCDownlinkMessage WILCO
12. CPDLC-message Indication
CPDLC Message = WILCO
TIME

Figure 4.7-12. CPDLC Message Exchanges, Logical Acknowledgments Prohibited

4.7.2.13 *Logical Acknowledgments Are Prohibited, Resulting ERROR When Requested*

4.7.2.13.1 Figure 4.7-12 illustrates the primitives and parameters of the CPDLC-start, CPDLC-message, D-START, and D-DATA services when a CPDLC-ground-user requests a CPDLC dialogue and includes a CPDLC message indicating that logical acknowledgments are prohibited (UM233) and a clearance element requiring an operational response (UM20). (The CPDLC message consists of two message elements.) The CPDLC-air-user accepts the dialogue request. The CPDLC SARPs do not permit a CPDLC message in a CPDLC start response when the request is accepted. Thus the CPDLC-message service is used to respond to the CPDLC message contained in the CPDLC-start request.

4.7.2.13.2 Furthermore, the CPDLC-air-user indicates that a logical acknowledgment is required on the WILCO. The SARPs require that an ERROR message be generated by the CPDLC-ground-user in response to any message sent by the CPDLC-air-user requiring a logical acknowledgment.

4.7.2.13.3 The CPDLC SARPs also require that whenever an ERROR message is sent in response to a given message, the received message is disregarded. Thus, this scenario depicts the CPDLC-air-user re-sending the WILCO message. An ERROR message is a “closure” message (closure is defined in 2.3.7 of the CPDLC SARPs), so the re-sent WILCO does not have response number that allows it to be automatically paired with its corresponding clearance.

4.7.2.13.4 The CPDLC SARPs do not require, nor do they prohibit, a CPDLC-air-user from responding with an ERROR message if a CPDLC-ground-user sends a message requiring a logical acknowledgment in a regime when the ground has prohibited its use.

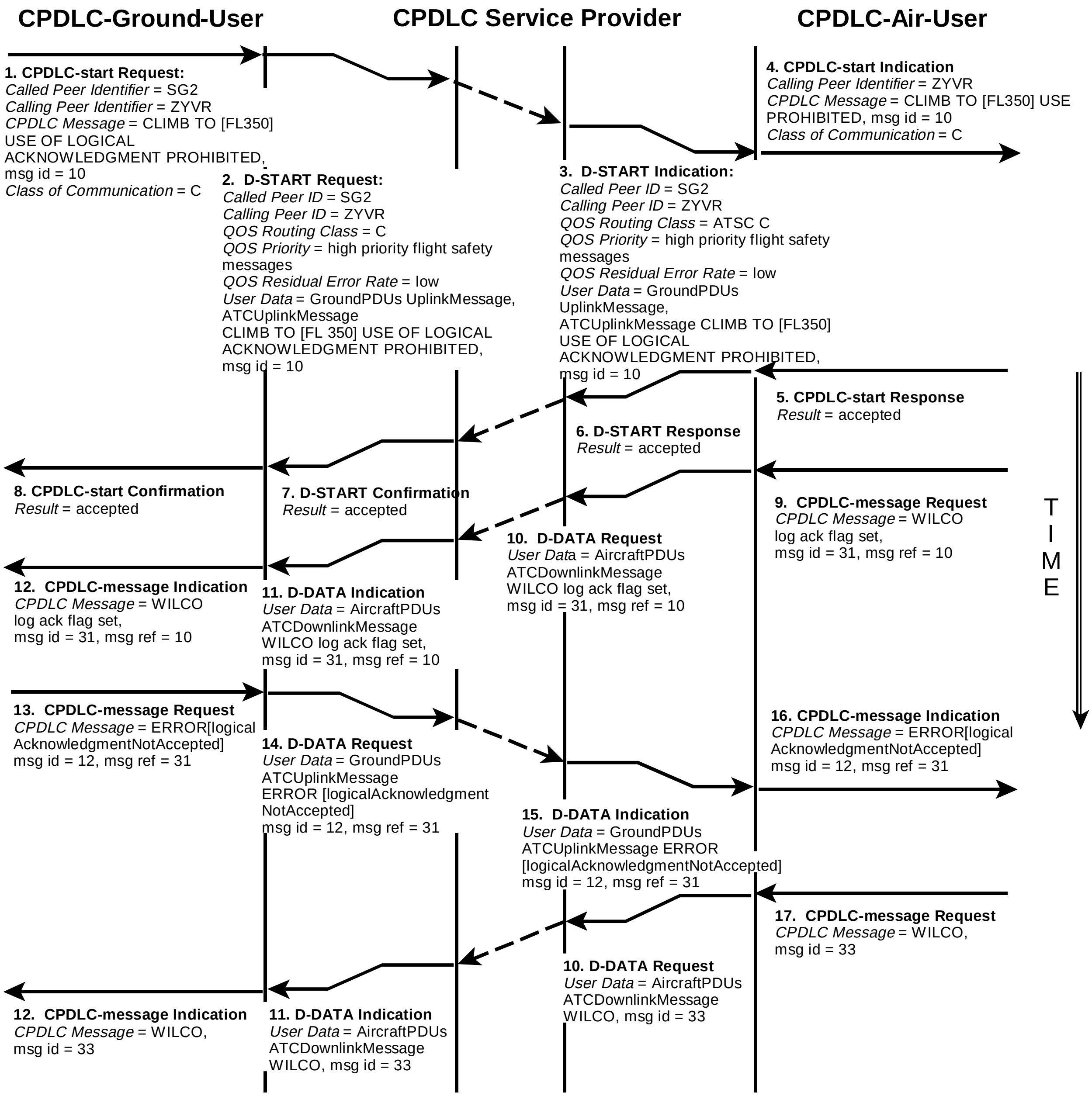

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-start Request:
Called Peer Identifier = SG2
Calling Peer Identifier = ZYVR
CPDLC Message = CLIMB TO [FL350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED, msg id = 10
Class of Communication = C
2. D-START Request:
Called Peer ID = SG2
Calling Peer ID = ZYVR
QOS Routing Class = C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage, ATCUplinkMessage CLIMB TO [FL 350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED, msg id = 10
3. D-START Indication:
Called Peer ID = SG2
Calling Peer ID = ZYVR
QOS Routing Class = ATSC C
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs UplinkMessage, ATCUplinkMessage CLIMB TO [FL350] USE OF LOGICAL ACKNOWLEDGMENT PROHIBITED, msg id = 10
4. CPDLC-start Indication
Calling Peer Identifier = ZYVR
CPDLC Message = CLIMB TO [FL350] USE PROHIBITED, msg id = 10
Class of Communication = C
5. CPDLC-start Response
Result = accepted
6. D-START Response
Result = accepted
7. D-START Confirmation
Result = accepted
8. CPDLC-start Confirmation
Result = accepted
9. CPDLC-message Request
CPDLC Message = WILCO log ack flag set, msg id = 31, msg ref = 10
10. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage WILCO log ack flag set, msg id = 31, msg ref = 10
11. D-DATA Indication
User Data = AircraftPDUs ATCDownlinkMessage WILCO log ack flag set, msg id = 31, msg ref = 10
12. CPDLC-message Indication
CPDLC Message = WILCO log ack flag set, msg id = 31, msg ref = 10
13. CPDLC-message Request
CPDLC Message = ERROR[logical AcknowledgmentNotAccepted] msg id = 12, msg ref = 31
14. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage ERROR [logicalAcknowledgment NotAccepted] msg id = 12, msg ref = 31
15. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage ERROR [logicalAcknowledgmentNotAccepted] msg id = 12, msg ref = 31
16. CPDLC-message Indication
CPDLC Message = ERROR[logical AcknowledgmentNotAccepted] msg id = 12, msg ref = 31
17. CPDLC-message Request
CPDLC Message = WILCO, msg id = 33
10. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage WILCO, msg id = 33
11. D-DATA Indication
User Data = AircraftPDUs ATCDownlinkMessage WILCO, msg id = 33
12. CPDLC-message Indication
CPDLC Message = WILCO, msg id = 33
TIME

Figure 4.7-13. CPDLC Message Exchanges, Logical Acknowledgments Prohibited

4.7.2.14 *Message Exchanges When Received Messages Requires Response, But is Found In Error*

4.7.2.14.1 Figure 4.7-14 illustrates the primitives and parameters of the CPDLC-message and D-DATA services when a CPDLC-ground-user sends a CPDLC message requiring a logical acknowledgment and an operational response. The CPDLC-air-user finds the message in error. This scenario illustrates what happens when uplink message element 33 is sent. The SARPs do not permit this message to be sent (so the CPDLC-ground-user is non-SARPs-compliant in this example), and require the CPDLC-air-user to issue an ERROR if this message element is received as a part of any CPDLC message. Operationally sending of *Reserved* message elements is considered an error rather than an abort condition. Other than decoding errors detected by a receiving ASE, any CPDLC message checking is incumbent on the CPDLC-user.0

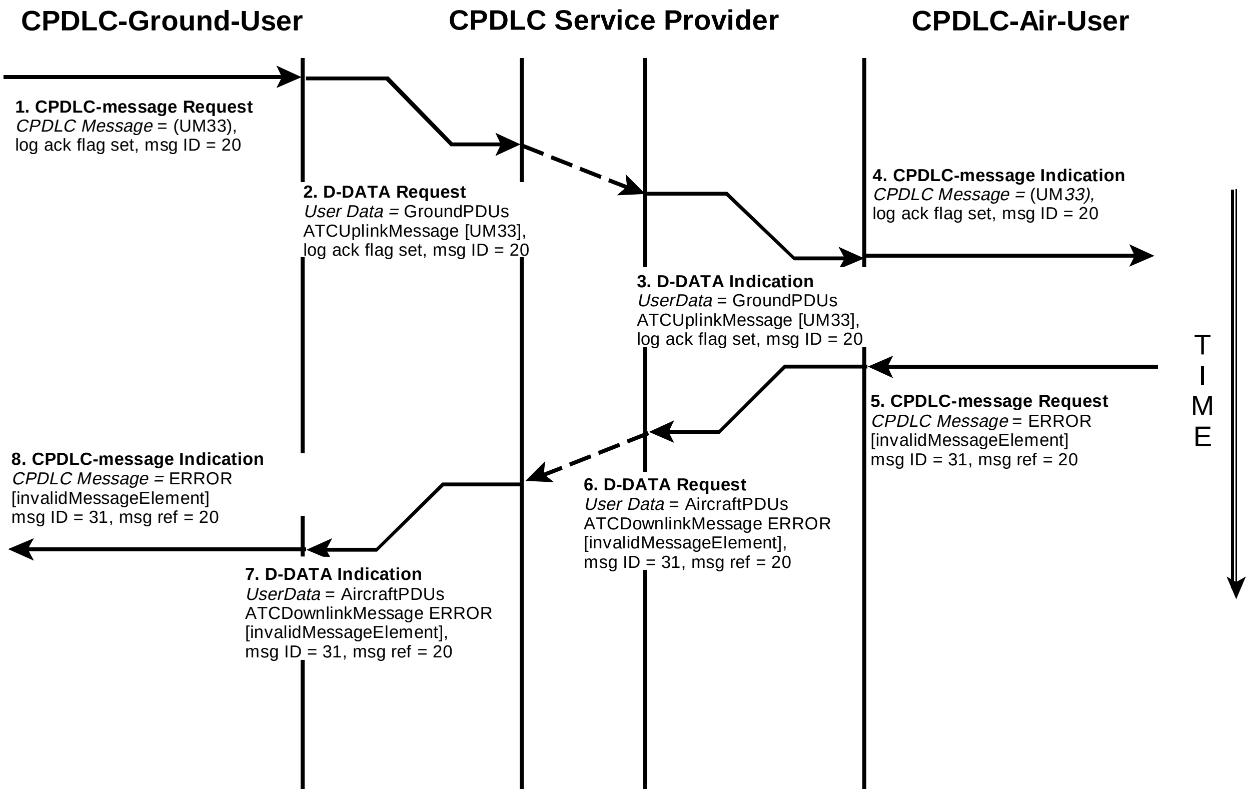


Figure 4.7-14. CPDLC Message Exchanges, Response Required, Error Detected

4.7.2.15 *Message Exchanges When Received Message Does Not Require Response, But is Found In Error*

4.7.2.15.1 Figure 4.7-15 illustrates the primitives and parameters of the CPDLC-message and D-DATA services when a CPDLC-air-user sends a CPDLC message requiring a logical acknowledgment and an operational response. The CPDLC-ground-user finds the message in error. This scenario illustrates what happens when downlink message element 108 is sent and the CPDLC-ground-user is unable to process the message due to insufficient resources. In this scenario the message (DM108) does not require an operational response, nor a logical acknowledgment. In this situation an ERROR response is sent, but it does not include a response identifier as in the previous example. Other than decoding errors detected by a receiving ASE, any CPDLC message checking is incumbent on the CPDLC-user.

4.7.2.15.2 Also note in this example that the CPDLC message sent by the CPDLC-air-user has the message identification number 31. The CPDLC message sent by the CPDLC-ground-user also has the message identifier 31. This is purely coincidental, and is operationally possible, since air and ground message identification numbers are totally independent. If in this example, the CPDLC-air-user had required a logical acknowledgment (not depicted), then the ERROR message generated by the CPDLC-ground-user would have had a message

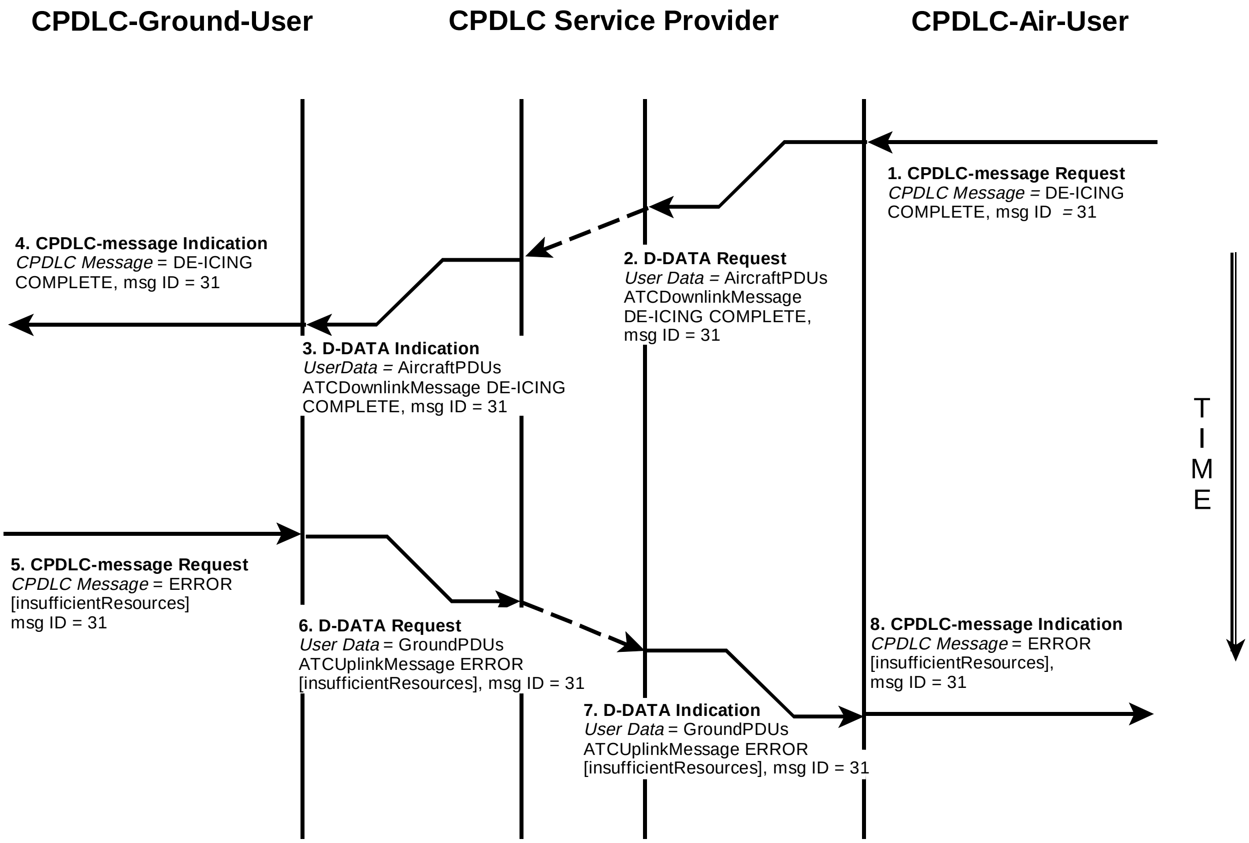

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-message Request
CPDLC Message = DE-ICING COMPLETE, msg ID = 31
2. D-DATA Request
User Data = AircraftPDUs ATCDownlinkMessage DE-ICING COMPLETE, msg ID = 31
3. D-DATA Indication
UserData = AircraftPDUs ATCDownlinkMessage DE-ICING COMPLETE, msg ID = 31
4. CPDLC-message Indication
CPDLC Message = DE-ICING COMPLETE, msg ID = 31
5. CPDLC-message Request
CPDLC Message = ERROR [insufficientResources] msg ID = 31
6. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage ERROR [insufficientResources], msg ID = 31
7. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage ERROR [insufficientResources], msg ID = 31
8. CPDLC-message Indication
CPDLC Message = ERROR [insufficientResources], msg ID = 31
T
I
M
E

Figure 4.7-15. CPDLC Message Exchanges, No Response Required, Error Detected

identifier of 31 (coincidental) and a message reference number of 31 (required to match the associated identification number to which the response is associated).

4.7.2.16 ***CPDLC-end, No Included Message, No Open Downlinks***

4.7.2.16.1 Figure 4.7-16 illustrates the primitives and parameters of the CPDLC-end and D-END services when a CDA CPDLC-ground-user sends a CPDLC end request without any CPDLC message and the CPDLC-air-user does not have any downlink open CPDLC messages. This scenario also applies to the situation when a CPDLC message is included in the end request that does not require any response (including not requiring a logical acknowledgment) and the CPDLC-air-user does not have any downlink open messages. The SARPs require that the CPDLC-air-user accept the end request in both of these cases. Uplink open messages are not a factor in the CPDLC-end service.

4.7.2.16.2 A CPDLC-end request can only be invoked by a CDA CPDLC-ground-user. The CPDLC-air-user must confirm that a CPDLC-end-indication is from a CDA. If a CPDLC-end indication is received from CPDLC-ground-user that is not the CDA, the SARPs require that the CPDLC-air-user reject the end and include the CPDLC message NOT CURRENT DATA AUTHORITY.

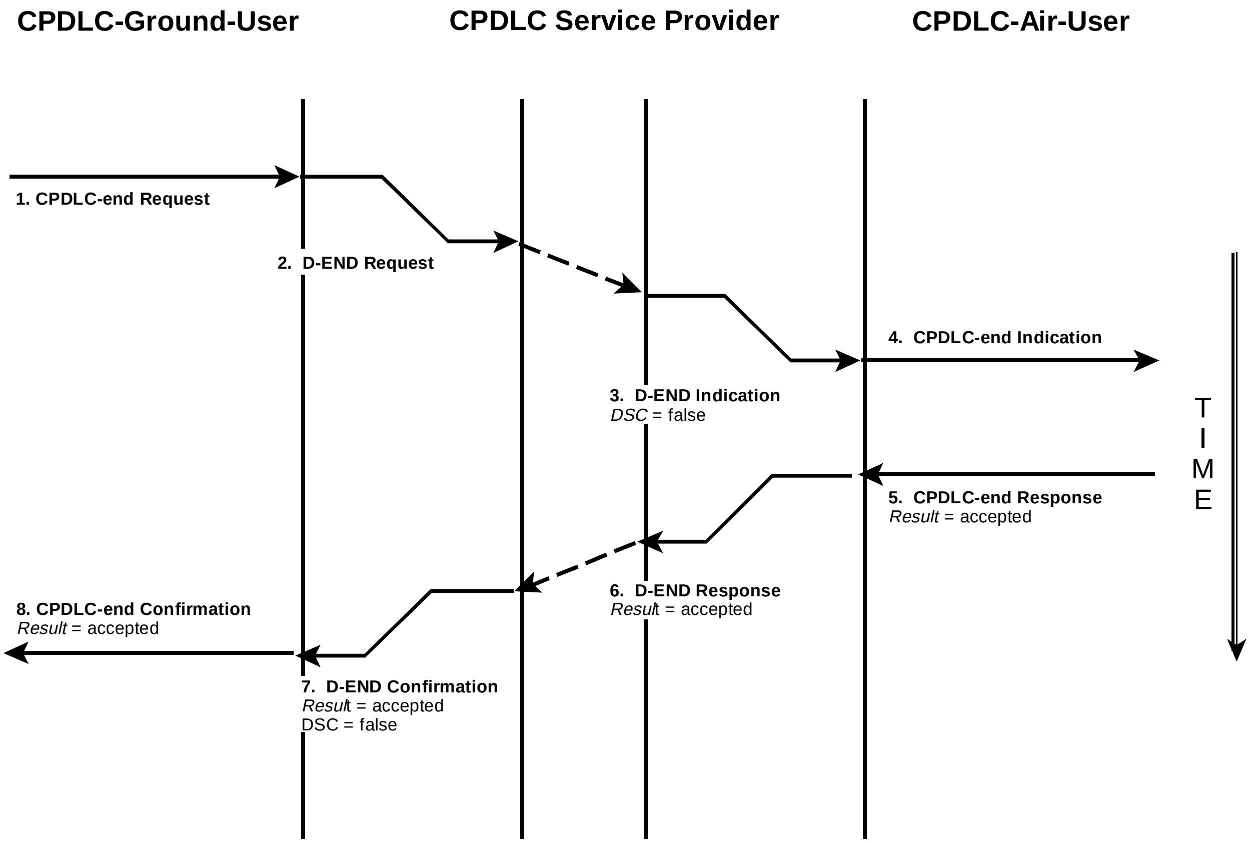


Figure 4.7-16. CPDLC-end, No Message Included, No Downlink Open Messages

4.7.2.17 ***CPDLC-end, Open Downlink(s)***

4.7.2.17.1 Figure 4.7-17 illustrates the primitives and parameters of the CPDLC-message, CPDLC-end, D-DATA, and D-END services when a CDA CPDLC-ground-user sends a CPDLC end request without a CPDLC message and the CPDLC-air-user does has open downlink CPDLC messages (an unanswered request is illustrated). This scenario also applies any CPDLC-end issued by the CDA when there are open downlink messages. The SARPs require that the CPDLC-air-user answer the end request (an “accept” or a “reject”), but do not constrain which of the two responses is given. This scenario shows the CPDLC-air-user rejecting the end request.

4.7.2.17.2 The CPDLC-ground-user could have included a closure response to the open downlink message shown in this scenario (e,g,, UNABLE DUE TO TRAFFIC (UM0 plus UM166) in the CPDLC-end request (not depicted)). The CPDLC-air user would no longer have open downlinks and would be required to accept the end request.

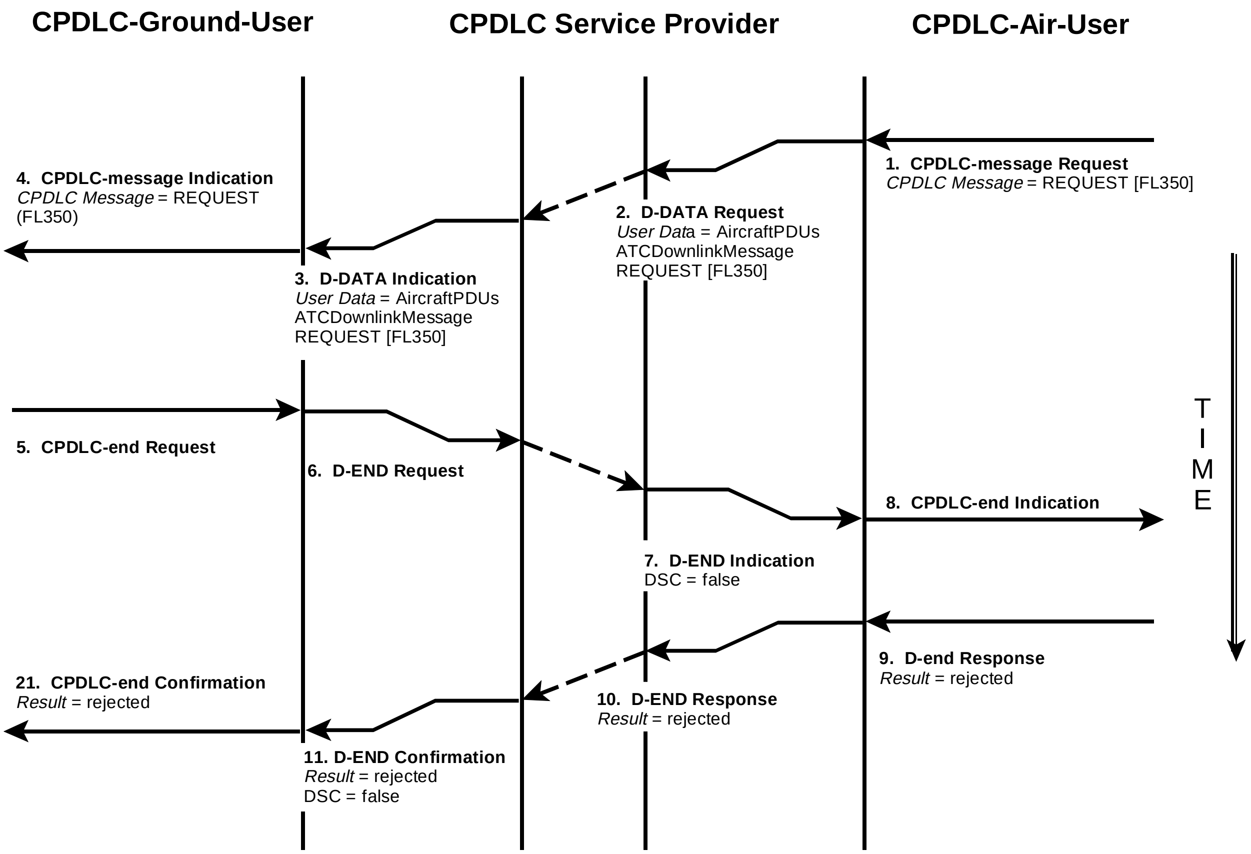


Figure 4.7-17. CPDLC-end, Open Downlink Messages

4.7.2.18 *Message in CPDLC-End Requires Responses, No Open Downlink Messages*

4.7.2.18.1 Figure 4.7-18 illustrates the primitives and parameters of the CPDLC-end, CPDLC-message, D-DATA, and D-END services when a CPDLC-ground-user sends a CPDLC end with a CPDLC message requiring a logical acknowledgment and operational response and the CPDLC-air-user does not have any open downlink CPDLC messages. Open plink messages are not a factor in the CPDLC-end service. This scenario shows an error-free situation.

4.7.2.18.2 The SARPs do not permit a CPDLC-ground-user to send messages once a CPDLC-end request has been issued, including a logical acknowledgment, until after the receipt of the CPDLC-end confirmation. When a CPDLC-end confirmation is received as "accepted", the dialogue is terminated and there is no way to exchange any CPDLC messages with the aircraft until a new CPDLC dialogue is established. When a CPDLC-end confirmation is received as "rejected", the given dialogue remains in place, and CPDLC messages can once again be exchanged. (The end is essentially canceled and must again be invoked to normally end the dialogue.)

4.7.2.18.3 Thus in this scenario, even though the CPDLC requested a logical acknowledgment on the WILCO message, the CPDLC-ground cannot provide one since the dialogue terminates. Even, if the CPDLC-air-user had used the CPDLC-message service for the WILCO, the

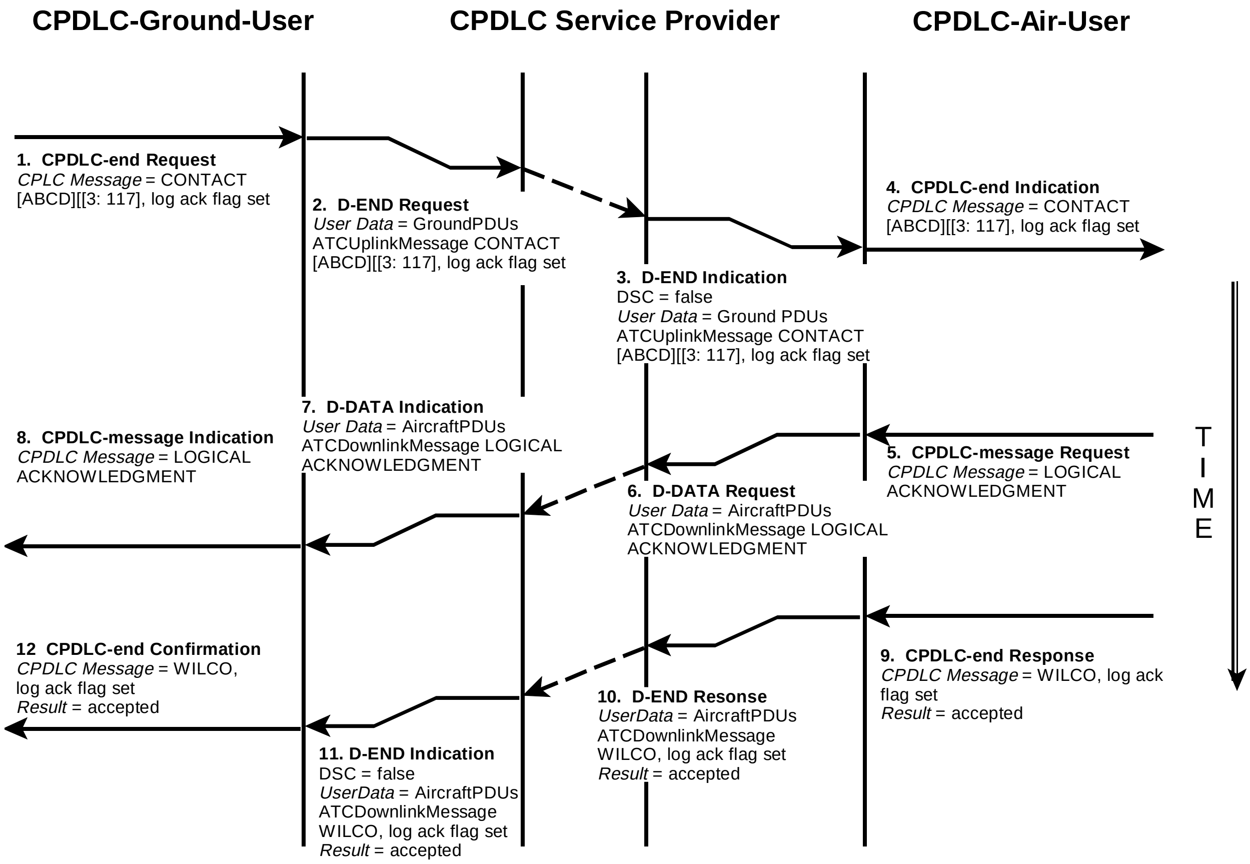


Figure 4.7-18. CPDLC-end, Message Requiring Response Included, No Downlink Open Messages

CPDLC-ground-user is prohibited from responding. In the case where there are no open downlink messages and the CPDLC-air-user is required to accept the end request, the CPDLC-ground-user cannot respond at all after issuing the CPDLC-end request.

4.7.2.18.4 If logical acknowledgments are required, all CPDLC messages could be sent using the CPDLC-message service; and the scenario depicted in Figure 4.7.16 could be used to terminate the CDA connection.

4.7.2.19 *DSC-end, No Included Message, No Open Uplinks*

4.7.2.19.1 Figure 4.7-19 illustrates the primitives and parameters of the DSC-end and D-END services when a CPDLC-air-user sends a DSC-end request without any CPDLC message and the CPDLC-ground-user does not have any open uplink CPDLC messages. This scenario also applies to the situation when a CPDLC message is included in the end request that does not require any response (including not requiring a logical acknowledgment) and the CPDLC-ground-user does not have any open uplink messages. The SARPs require that the CPDLC-ground-user accept the end request in both of these cases. Open downlink messages are not a factor in the DSC-end service.

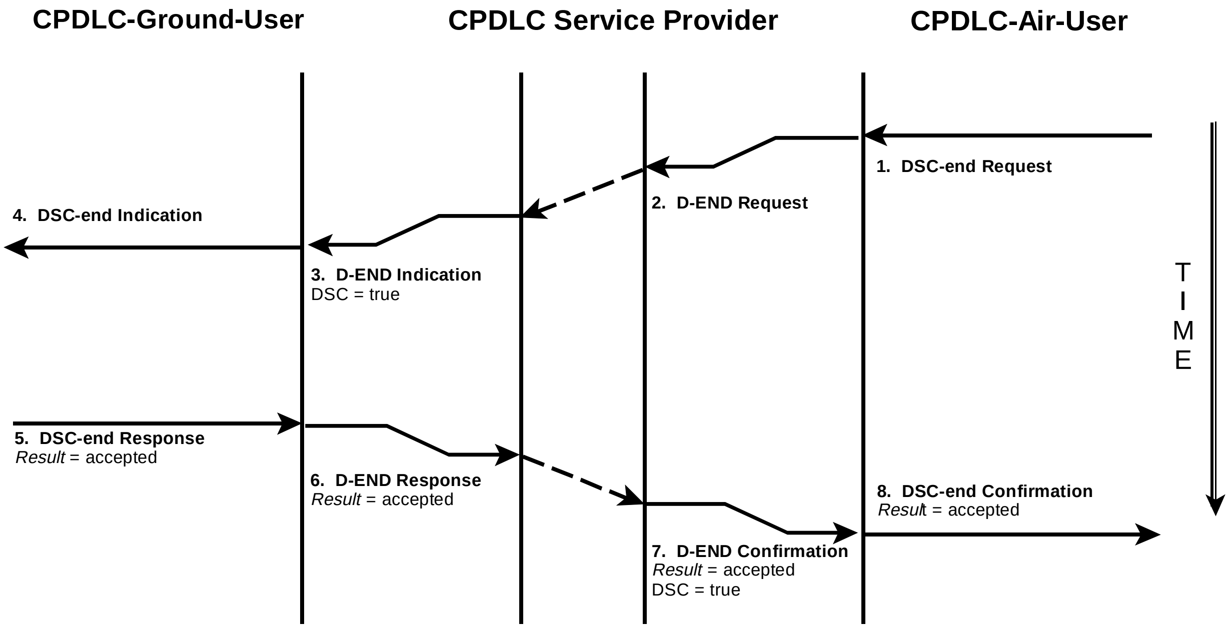


Figure 4.7-19. DSC-end, No message Included, No Uplink Open Messages

4.7.2.20 ***DSC-end, Open Uplinks***

4.7.2.20.1 Figure 4.7-20 illustrates the primitives and parameters of the CPDLC-message, DSC-end, D-DATA, and D-END services when a CPDLC-air-user sends a CPDLC-end request without a CPDLC message and the CPDLC-ground-user has open uplink CPDLC messages (an unanswered EXPECT message is illustrated). This scenario also applies to any DSC-end issued by the CPDLC-air-user when there are open uplink messages. The SARPs require that the CPDLC-ground-user answer the end request (an "accept" or a "reject"), but do not constrain which of the two responses is given. This scenario shows the CPDLC-ground-user accepting the end request. (For this particular message, it was just a "heads-up" so an actual clearance will be issued when the DDA becomes the CDA, so the CPDLC-ground-user accepts the end request in spite of open uplink messages.)

4.7.2.20.2 The CPDLC-air-user could have included a closure response to the open uplink message shown in this scenario (i.e., ROGER (DM3)) in the DSC-end request (not depicted). The CPDLC-ground-user would no longer have open uplinks and would be required to accept the end request.

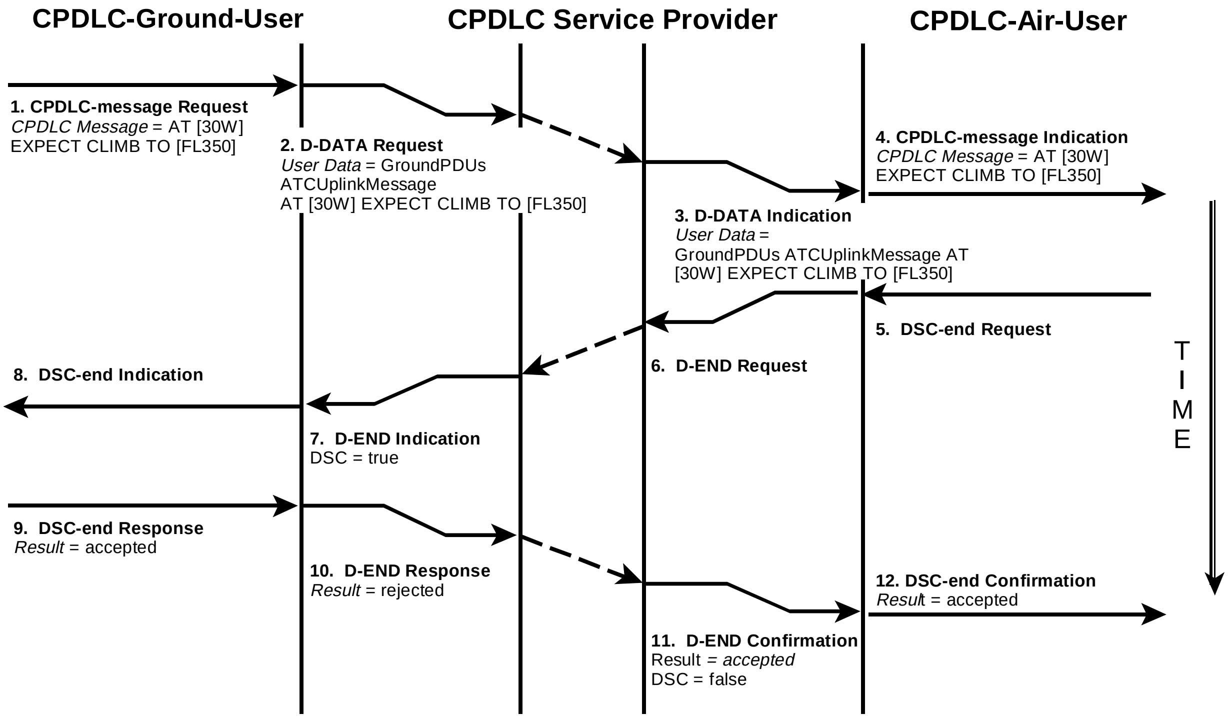


Figure 4.7-20. CPDLC-end, Open Downlink Messages

4.7.2.21 *Message in DSC-End Requires Responses, No Open Uplink Messages*

4.7.2.21.1 Figure 4.7-21 illustrates the primitives and parameters of the DSC-end, CPDLC-message, D-DATA, and D-END services when a CPDLC-air-user sends a DSC-end with a CPDLC message requiring a logical acknowledgment and operational response and the CPDLC-ground-user does not have any open uplink CPDLC messages. Downlink open messages are not a factor in the DSC-end service. This scenario shows an error-free situation.

4.7.2.21.2 The SARPs do not permit a CPDLC-air-user to send messages once a CPDLC-end request has been issued, including a logical acknowledgment, until after the receipt of the DSC-end confirmation. When a DSC-end confirmation is received as "accepted", the dialogue is terminated and there is no way to exchange any CPDLC messages with the aircraft until a new DSC dialogue is established. When a DSC-end confirmation is received as "rejected", the given dialogue remains in place, and CPDLC messages can once again be exchanged. (The end is essentially canceled an must again be invoked to normally end the dialogue.)

4.7.2.21.3 Thus in this scenario, even though the CPDLC-ground requested a logical acknowledgment on the UNABLE message, the CPDLC-air cannot provide one since the dialogue terminates. Even if the CPDLC-ground-user had used the CPDLC-message service for the UNABLE,

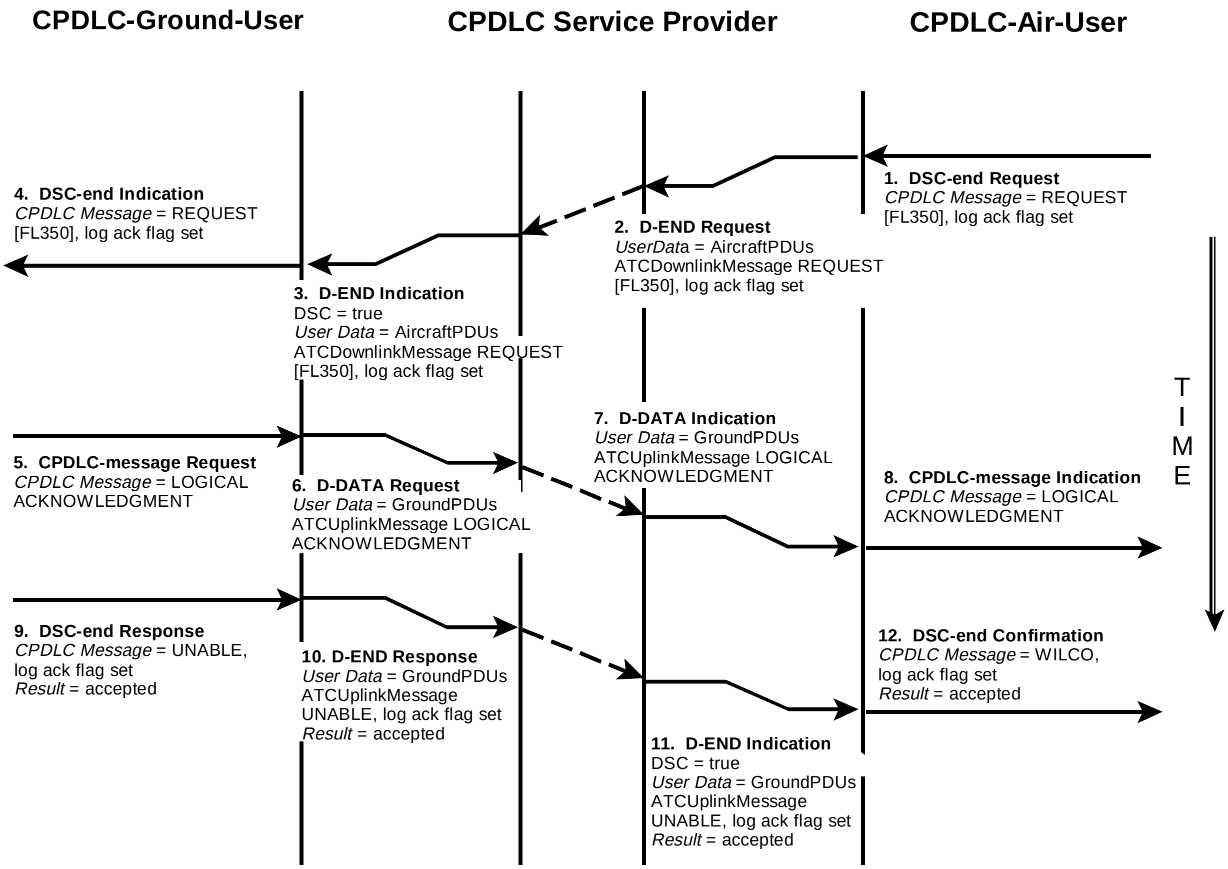

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. DSC-end Request
CPDLC Message = REQUEST [FL350], log ack flag set
2. D-END Request
UserData = AircraftPDUs ATCDownlinkMessage REQUEST [FL350], log ack flag set
3. D-END Indication
DSC = true
User Data = AircraftPDUs ATCDownlinkMessage REQUEST [FL350], log ack flag set
4. DSC-end Indication
CPDLC Message = REQUEST [FL350], log ack flag set
5. CPDLC-message Request
CPDLC Message = LOGICAL ACKNOWLEDGMENT
6. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT
7. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage LOGICAL ACKNOWLEDGMENT
8. CPDLC-message Indication
CPDLC Message = LOGICAL ACKNOWLEDGMENT
9. DSC-end Response
CPDLC Message = UNABLE, log ack flag set
Result = accepted
10. D-END Response
User Data = GroundPDUs ATCUplinkMessage UNABLE, log ack flag set
Result = accepted
11. D-END Indication
DSC = true
User Data = GroundPDUs ATCUplinkMessage UNABLE, log ack flag set
Result = accepted
12. DSC-end Confirmation
CPDLC Message = WILCO, log ack flag set
Result = accepted
TIME

Figure 4.7-21. DSC-end, Message Requiring Response Included, No Open Uplink Messages

the CPDLC-ground-user is prohibited from responding. In the case where there are no open downlink messages and the CPDLC-air-user is required to accept the end request, the CPDLC-ground-user cannot respond at all after issuing the CPDLC-end request.

4.7.2.21.4 If logical acknowledgments are required, all CPDLC messages could be sent using the CPDLC-message service; and the scenario depicted in Figure 4.7-19 could be used to terminate the DSC connection.

4.7.2.22 *CPDLC-forward Service, Receiver Does Not Support Forward Service*

4.7.2.22.1 Figure 4.7-22 illustrates the primitives and parameters of the CPDLC-forward and D-START services when a CPDLC-ground-user uses the CPDLC-forward request to forward a CPDLC message to another CPDLC-ground system and the receiving ground system does not support the forward service.

4.7.2.22.2 A ground system must, at a minimum, recognize and reject the forward request even if it does not support the forward service. The SARPs do not require that a ground system ever be an initiator of the CPDLC-forward service.

4.7.2.23 *CPDLC-forward Service, Version Numbers Not Equal*

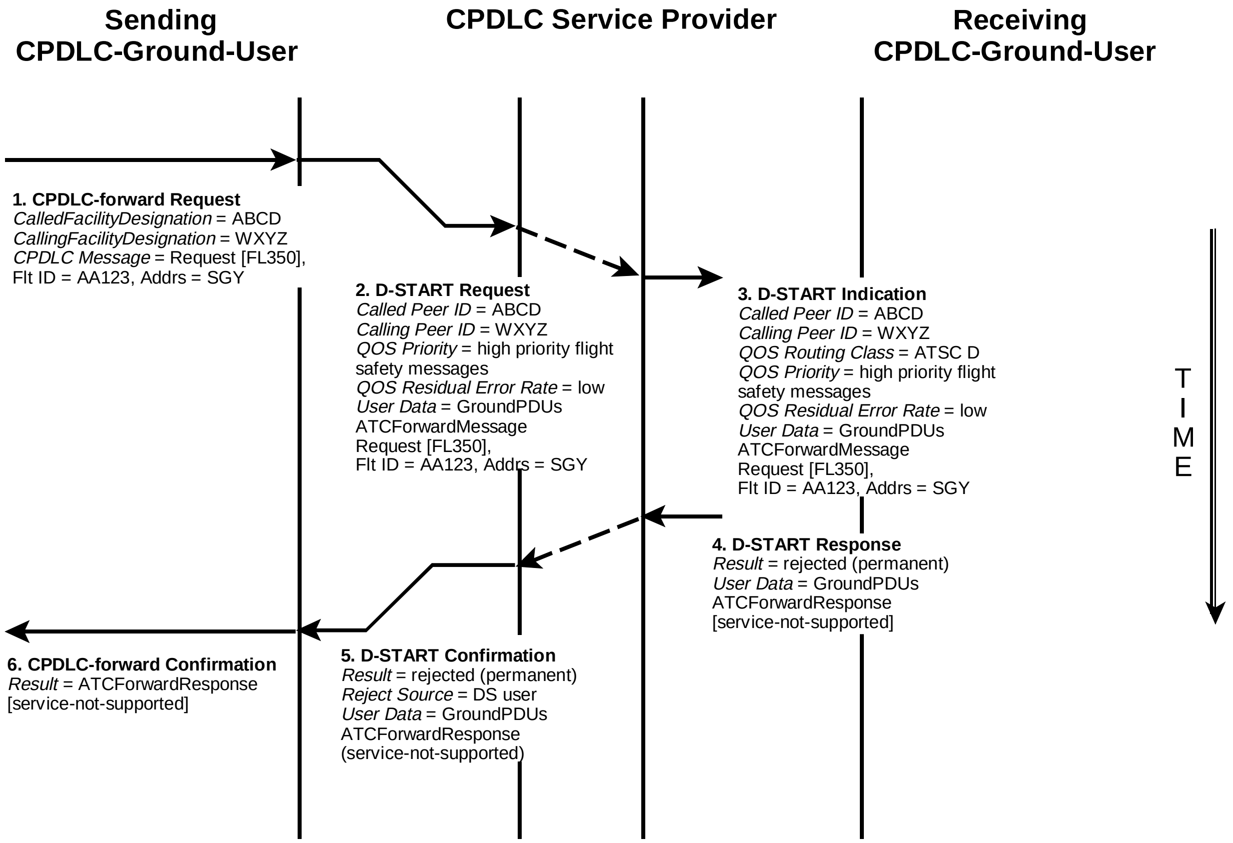

Sending CPDLC-Ground-User
CPDLC Service Provider
Receiving CPDLC-Ground-User
1. CPDLC-forward Request
CalledFacilityDesignation = ABCD
CallingFacilityDesignation = WXYZ
CPDLC Message = Request [FL350], Flt ID = AA123, Addrs = SGY
2. D-START Request
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage Request [FL350], Flt ID = AA123, Addrs = SGY
3. D-START Indication
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage Request [FL350], Flt ID = AA123, Addrs = SGY
T I M E
4. D-START Response
Result = rejected (permanent)
User Data = GroundPDUs ATCForwardResponse [service-not-supported]
5. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = GroundPDUs ATCForwardResponse (service-not-supported)
6. CPDLC-forward Confirmation
Result = ATCForwardResponse [service-not-supported]

Figure 4.7-22. CPDLC-forward, Receiving Ground Does Not Support Forward Service

4.7.2.23.1 Figure 4.7-23 illustrates the primitives and parameters of the CPDLC-forward and D-START services when a CPDLC-ground-user uses the CPDLC-forward request to forward a CPDLC message to another CPDLC-ground system and the receiving ground system does not match the version of the initiating ground system.

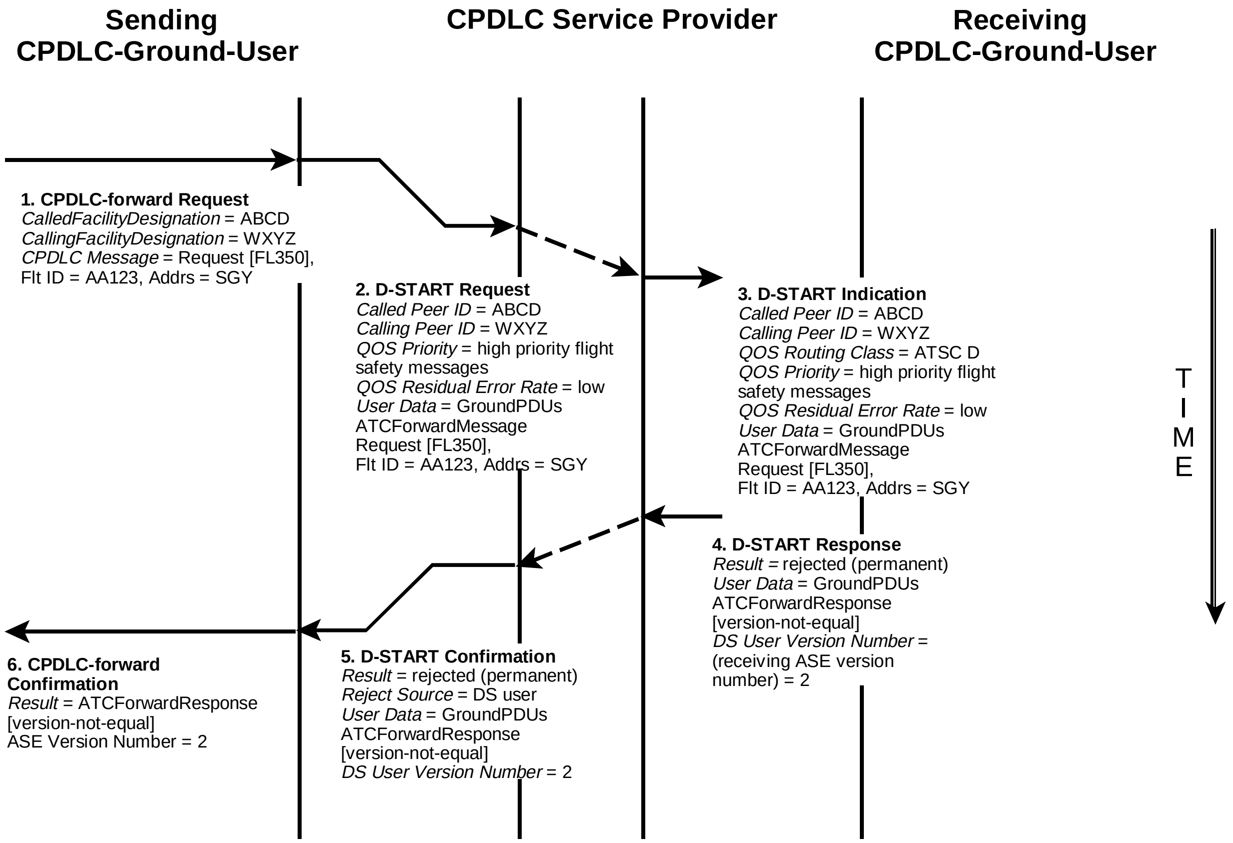

Sending
CPDLC-Ground-User
CPDLC Service Provider
Receiving
CPDLC-Ground-User
1. CPDLC-forward Request
CalledFacilityDesignation = ABCD
CallingFacilityDesignation = WXYZ
CPDLC Message = Request [FL350],
Flt ID = AA123, Addrs = SGY
2. D-START Request
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs
ATCForwardMessage
Request [FL350],
Flt ID = AA123, Addrs = SGY
3. D-START Indication
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs
ATCForwardMessage
Request [FL350],
Flt ID = AA123, Addrs = SGY
T
I
M
E
4. D-START Response
Result = rejected (permanent)
User Data = GroundPDUs
ATCForwardResponse
[version-not-equal]
DS User Version Number =
(receiving ASE version number) = 2
5. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = GroundPDUs
ATCForwardResponse
[version-not-equal]
DS User Version Number = 2
6. CPDLC-forward Confirmation
Result = ATCForwardResponse
[version-not-equal]
ASE Version Number = 2

Figure 4.7-23. CPDLC-forward, Incompatible Versions

4.7.2.24 ***CPDLC-forward Service, Service Successful***

4.7.2.24.1 Figure 4.7-24 illustrates the primitives and parameters of the CPDLC-forward and D-START services when a CPDLC-ground-user uses the CPDLC-forward request to forward a CPDLC message to another CPDLC-ground system and the message forwarding is successful.

4.7.2.24.2 CPDLC-ground forwarding is "one-shot", so even if the service is successful, as illustrated, the forward request is rejected and the dialogue terminated upon forward confirmation.

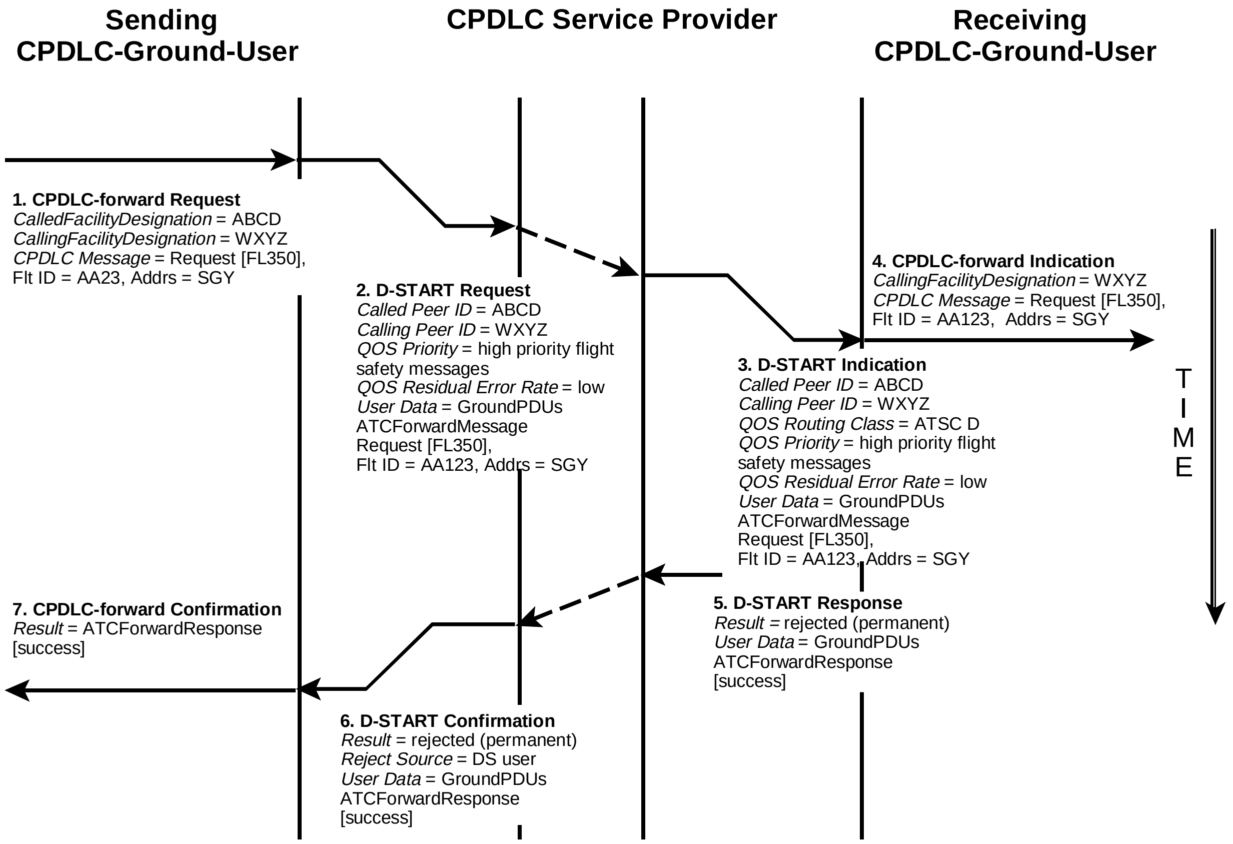

Sending CPDLC-Ground-User
CPDLC Service Provider
Receiving CPDLC-Ground-User
1. CPDLC-forward Request
CalledFacilityDesignation = ABCD
CallingFacilityDesignation = WXYZ
CPDLC Message = Request [FL350],
Flt ID = AA23, Addrs = SGY
2. D-START Request
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage Request [FL350], Flt ID = AA123, Addrs = SGY
3. D-START Indication
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage Request [FL350], Flt ID = AA123, Addrs = SGY
4. CPDLC-forward Indication
CallingFacilityDesignation = WXYZ
CPDLC Message = Request [FL350],
Flt ID = AA123, Addrs = SGY
5. D-START Response
Result = rejected (permanent)
User Data = GroundPDUs ATCForwardResponse [success]
6. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = GroundPDUs ATCForwardResponse [success]
7. CPDLC-forward Confirmation
Result = ATCForwardResponse [success]
TIME

Figure 4.7-24. CPDLC-forward, Service Successful

4.7.2.25 *Ground Forwarding Successful Both Sides Sender/Receiver*

4.7.2.25.1 Figure 4.7-25 illustrates the primitives and parameters of the CPDLC-forward and D-START services when a CPDLC-ground-user uses the CPDLC-forward request to forward a CPDLC message to another CPDLC-ground system and the message forwarding is successful and vise versa. Forwarding messages in both direction is operationally acceptable. In this case a given pair of ground systems can be both a receiver and an initiator concurrently. Each forward request establishes an independent connection. There is no two-way message forwarding over the same connection. The forward requests/responses are shown crossing. This could happen, but need not happen. The forward services could be totally sequential.

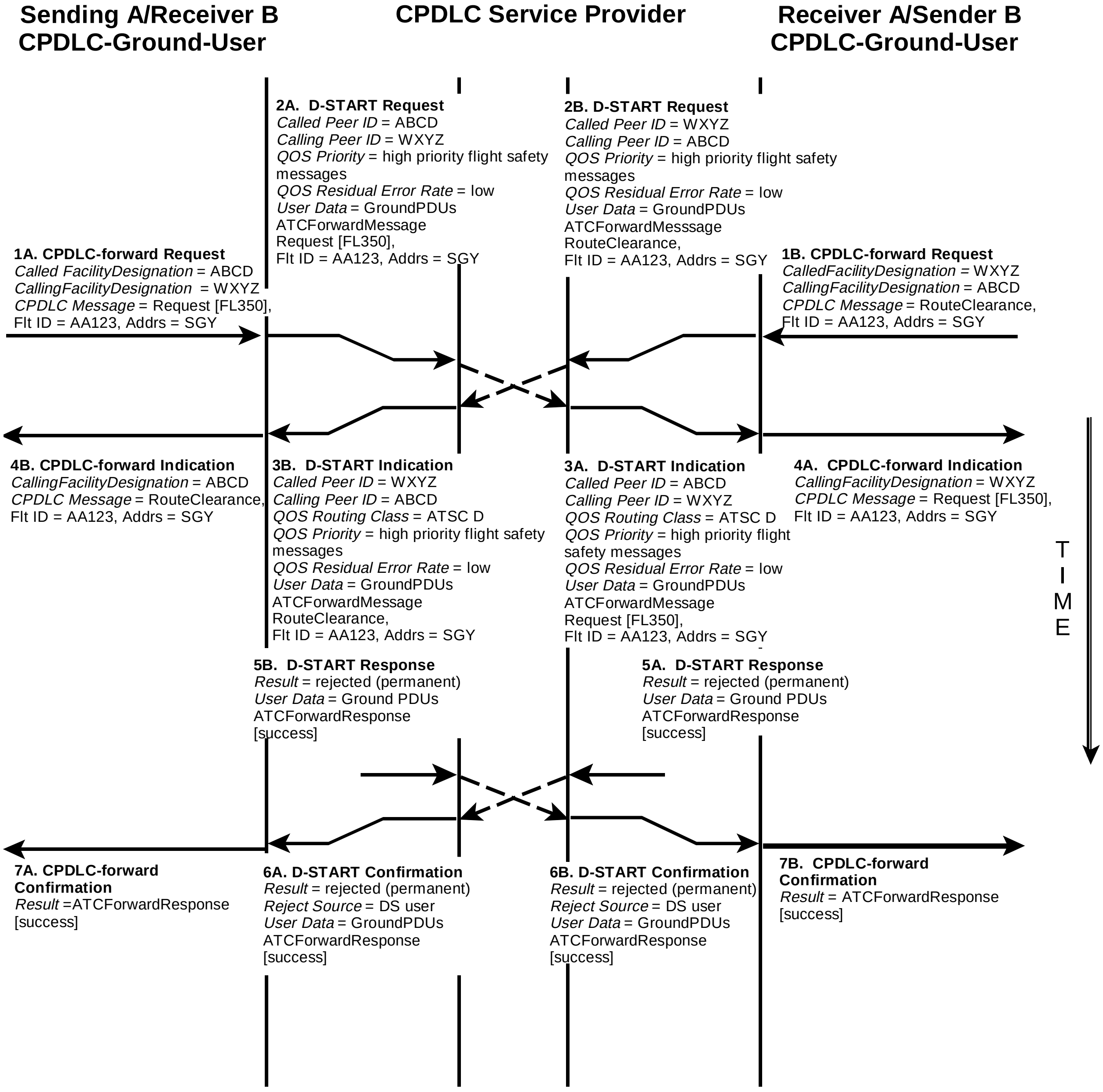

Sending A/Receiver B
CPDLC-Ground-User
CPDLC Service Provider
Receiver A/Sender B
CPDLC-Ground-User
2A. D-START Request
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage Request [FL350], Flt ID = AA123, Addrs = SGY
2B. D-START Request
Called Peer ID = WXYZ
Calling Peer ID = ABCD
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMesssage RouteClearance, Flt ID = AA123, Addrs = SGY
1A. CPDLC-forward Request
Called FacilityDesignation = ABCD
CallingFacilityDesignation = WXYZ
CPDLC Message = Request [FL350], Flt ID = AA123, Addrs = SGY
1B. CPDLC-forward Request
CalledFacilityDesignation = WXYZ
CallingFacilityDesignation = ABCD
CPDLC Message = RouteClearance, Flt ID = AA123, Addrs = SGY
4B. CPDLC-forward Indication
CallingFacilityDesignation = ABCD
CPDLC Message = RouteClearance, Flt ID = AA123, Addrs = SGY
3B. D-START Indication
Called Peer ID = WXYZ
Calling Peer ID = ABCD
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage RouteClearance, Flt ID = AA123, Addrs = SGY
3A. D-START Indication
Called Peer ID = ABCD
Calling Peer ID = WXYZ
QOS Routing Class = ATSC D
QOS Priority = high priority flight safety messages
QOS Residual Error Rate = low
User Data = GroundPDUs ATCForwardMessage Request [FL350], Flt ID = AA123, Addrs = SGY
4A. CPDLC-forward Indication
CallingFacilityDesignation = WXYZ
CPDLC Message = Request [FL350], Flt ID = AA123, Addrs = SGY
T I M E
5B. D-START Response
Result = rejected (permanent)
User Data = Ground PDUs ATCForwardResponse [success]
5A. D-START Response
Result = rejected (permanent)
User Data = Ground PDUs ATCForwardResponse [success]
7A. CPDLC-forward Confirmation
Result =ATCForwardResponse [success]
6A. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = GroundPDUs ATCForwardResponse [success]
6B. D-START Confirmation
Result = rejected (permanent)
Reject Source = DS user
User Data = GroundPDUs ATCForwardResponse [success]
7B. CPDLC-forward Confirmation
Result = ATCForwardResponse [success]

Figure 4.7-25. CPDLC-forward, Service Successful Both Ground Systems Concurrently an Initiator and Receiver

4.7.2.26 ***CPDLC-User Abort***

4.7.2.26.1 Figure 4.7-26 illustrates the primitives and parameters of the CPDLC-user-abort and D-ABORT services when a CPDLC-ground-user-user issues an abort. The scenario is basically (the converse) the same when the CPDLC-air-user issues an abort (not depicted).

4.7.2.26.2 This scenario is typical of how the CPDLC-user-abort service works when both CPDLC-users are active. Whenever a given user is not active the CPDLC-abort-indication is not provided to that user.

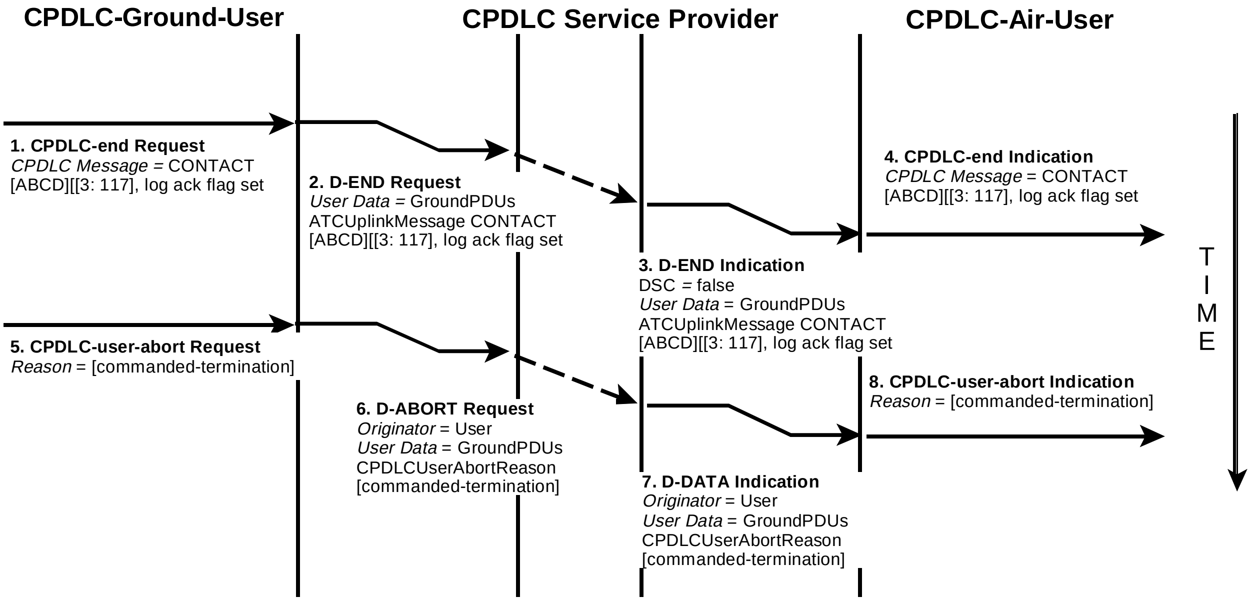


Figure 4.7-26. CPDLC-user abort: Both Users Active

4.7.2.27 *CPDLC-Provider-Abort, ASE Aborts*

4.7.2.27.1 Figure 4.7-27 illustrates the primitives and parameters of the CPDLC-message, CPDLC-provider-abort, D-DATA, and D-ABORT services when a CPDLC-air-ASE aborts when detecting an invalid PDU in the User Data of a D-DATA Indication. Both CPDLC-users are active.

4.7.2.27.2 Note that upon receipt of a D-ABORT indication, a CPDLC-ASE determines whether to issue a CPDLC-user or CPDLC-provider abort based on the value of the D-ABORT *Originator* parameter. The value "user" results in a CPDLC-user-abort indication to its CPDLC-user. The value "provider" results in a CPDLC-provider-abort indication to its CPDLC-user.

4.7.2.27.3 This scenario is typical of how the D-ABORT initiated CPDLC-provider-abort service works when both CPDLC-users are active. Whenever a given user is not active the CPDLC-abort-indication is not provided to that user.

4.7.2.27.4 This scenario shows the abort sequentially after the CPDLC-message service. However, aborts can occurs at any time, and they may overtake or interrupt any currently on-going "normal" activity.

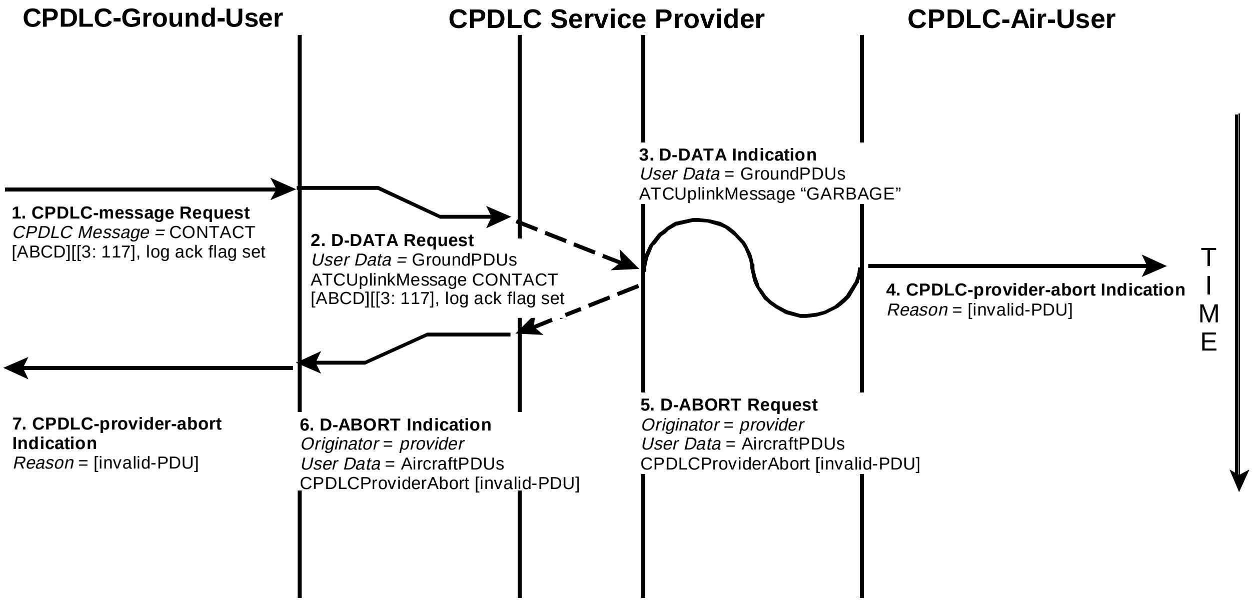

CPDLC-Ground-User
CPDLC Service Provider
CPDLC-Air-User
1. CPDLC-message Request
CPDLC Message = CONTACT [ABCD][[3: 117], log ack flag set
2. D-DATA Request
User Data = GroundPDUs ATCUplinkMessage CONTACT [ABCD][[3: 117], log ack flag set
3. D-DATA Indication
User Data = GroundPDUs ATCUplinkMessage "GARBAGE"
4. CPDLC-provider-abort Indication
Reason = [invalid-PDU]
5. D-ABORT Request
Originator = *provider*
User Data = AircraftPDUs CPDLCProviderAbort [invalid-PDU]
6. D-ABORT Indication
Originator = *provider*
User Data = AircraftPDUs CPDLCProviderAbort [invalid-PDU]
7. CPDLC-provider-abort Indication
Reason = [invalid-PDU]
TIME

Figure 4.7-27. CPDLC-ASE Aborts: Both Users Active

4.7.2.28 *CPDLC-Provider-Abort, Underlying Communication Service Aborts*

4.7.2.28.1 Figure 4.7-28 illustrates the primitives and parameters of the CPDLC-message, CPDLC-provider-abort, D-DATA, and D-P-ABORT services when the underlying communication service aborts. Both CPDLC-users are active D-P-ABORT indications always result in CPDLC-provider-abort indications to active CPDLC-users.

4.7.2.28.2 This scenario is typical of how the D-P-ABORT initiated CPDLC-provider-abort service works when both CPDLC-users are active. Whenever a given user is not active the CPDLC-abort-indication is not provided to that user.

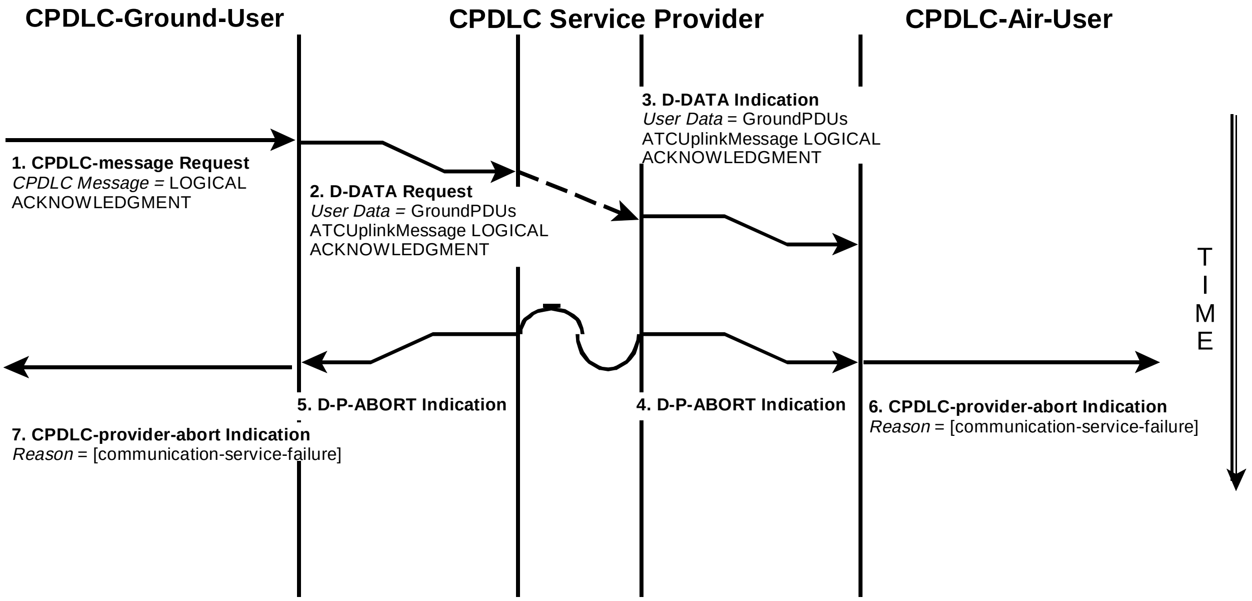


Figure 4.7-28. Underlying Communication Service Aborts: Both Users Active

4.8 Example Encoding

4.8.1 Encoding/Decoding Rules

4.8.1.1 Samples of PER encoded CPDLC messages are provided in this section. For each APDU, the value tree containing the values assigned to the APDU components is provided. Then, the octet string resulting of the PER encoding is dumped in hexadecimal.

4.8.1.2 The PER in this section has been calculated by hand and should be checked with a reference PER complier.

4.8.1.3 *CPDLC GroundPDUs [startup] without CPDLC Message*

Element	*Type*	*Value*	*Encoding*	*Comments*
GroundPDUs	CHOICE	startup	0 010	No extension used The CHOICE [2] is selected
UplinkMessage	CHOICE	noMessage	0xxx	The CHOICE [0] is selected

Hexadecimal view (1 octet): 20

4.8.1.4 *CPDLC AircraftPDUs [startdown] without CPDLC Message*

Element	*Type*	*Value*	*Encoding*	*Comments*
AircraftPDUs	CHOICE	startdown	0 10	No extension used The CHOICE [2] is selected
StartDownMessage	SEQUENCE			
mode			0	Default value 'cpdlc' is selected
startDownMessage	CHOICE	noMessage	0xxx	The CHOICE [0] is selected

Hexadecimal view (1 octet): 40

4.8.1.5 *CPDLC GroundPDUs [send]*

Common header (7 octets, 2 bit)

Element	*Type*	*Value*	*Encoding*	*Comments*
GroundPDUs	CHOICE	send	0 011	No extension used The CHOICE [3] is selected
ATCUplinkMessage	SEQUENCE			
Header	SEQUENCE		00	Default value 'notRequired' is selected
MessageIdNumber	INTEGER (0..63)	0	00.00000	
DateTime	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1997	000.0001	

Element	***Type***	***Value***	***Encoding***	***Comments***
Month	INTEGER (1..12)	11	1010.	11-1
Day	INTEGER (1..31)	2	00001	2-1
Timehhmmss	SEQUENCE			
Time	SEQUENCE			
Hours	INTEGER (0…23)	0	000.00	
Minutes	INTEGER (0…59)	0	000000.	
Seconds	INTEGER (0..59)	0	000000	
ElementsIds			00.1	1 element id
ATCUplinkMsgElementId	CHOICE		0 yyyyyy.yy	No extension used CHOICE [yyy] selected

- For Messages Uplink with contents **NULL** (0, 1, 2, 3, 4, 5, 33, 67, 72, 96, 107, 116, 124, 125, 126, 127, 131, 132, 133, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 154, 161, 162, 164, 165, 167, 168, 176, 177, 178, 179, 182, 189, 191, 193, 200, 201, 202, 211, 216, 217, 218, 222, 223, 224, 225, 227, 229, 230, 231, 232, 233, 234, 235, 236): no additional bits.

- Messages Uplink with contents **Level singleLevel** or **blockLevel** (6, 19, 20, 23, 34, 35, 36, 37, 38, 39, 40, 41128, 129, 148, 175,219, 220): add 11 bits or 21 bits. Messages Uplink with contents **LevelLevel** (30, 31, 32), add 20 bits.

Level	CHOICE	BlockLevel	1	CHOICE [1] is selected
LevelType (1/2)	CHOICE	LegDistance	0	CHOICE [0] is selected
LegDistance	CHOICE	LegDistanceMetric	1	CHOICE [1] is selected
LegDistanceMetric	INTEGER (1..128)	3	00000010	3-1
LevelType (2/2)	CHOICE	LegDistance	0	CHOICE [0] is selected
LegDistance	CHOICE	LegDistanceMetric	1	CHOICE [1] is selected
LegDistanceMetric	INTEGER (1..128)	3	00000010	3-1

- Messages Uplink with contents **Time** (7, 9, 11, 69, 71, 93, 226): add 11 bits. **TimeTime** add 22 bits.

Time	SEQUENCE			
Hours	INTEGER (0…23)	14	01110	
Minutes	INTEGER (0…59)	0	000000	

- Messages Uplink with contents **Position** (8, 10, 12, 68, 70, 74, 75, 87, 130, 155, 210, 228): add 54 bits

Position	CHOICE	Navaid	000	CHOICE [1] is selected
Navaid	SEQUENCE		1	Latlon is present
name	IA5String (1..4)	4 “ABCD”	100	Length = 4
latlon	SEQUENCE		11	Latitude present, Longitude present
latitude	SEQUENCE			

Position	CHOICE	Navaid	000	CHOICE [1] is selected
latitudeType	CHOICE	Latitude DMS	11	CHOICE [2] is selected
LatitudeDegreesMinutesSeconds	SEQUENCE			
LatitudeWholeDegree	INTEGER (0..89)	40°	0101000	
LatlonWholeMinutes	INTEGER (0..59)	40'	101000	
SecondsLatLon	INTEGER (0..59)	40"	101000	
LatitudeDirection	ENUMERATED (0..1)	North	0	
Longitude	SEQUENCE			
LongitudeType	CHOICE	LongitudeDMS	11	CHOICE [2] is selected
LongitudeDegreesMinutesSeconds	SEQUENCE			
LongitudeWholeDegrees	INTEGER (0..179)	40°	00101000	
LatlonWholeMinutes	INTEGER (0..59)	40'	101000	
SecondsLatLon	INTEGER (0..59)	40"	101000	
LongitudeDirection	ENUMERATED (0..1)	East	0	

- Messages Uplink with contents **TimeLevel** (13, 15, 17, 21, 24) or **LevelTime** (26, 28, 150, 129): add 22 bits.

TimeLevel	SEQUENCE			
Time				11 bits
Level (singleBlock)				11 bits

TimeLevel	SEQUENCE			
Level (singleBlock)				11 bits
Time				11 bits

- Messages Uplink with contents **PositionLevel** (14, 16, 22, 25, 42, 45, 46, 47, 48, 49, 92, 185, 186) or **LevelPosition** (27, 29, 90, 149, 209) : add 66 bits.

PositionLevel	SEQUENCE			
Position				54 bits
Level (singleBlock)				11 bits

LevelPosition	SEQUENCE			
Level (singleBlock)				11 bits
Position				54 bits

- Messages Uplink with contents **PositionLevelLevel** (50) : add 74 bits.

PositionLevel	SEQUENCE			
Position				54 bits
LevelLevel				20 bits

- Messages Uplink with contents **PositionTime** (51, 52, 53) : add 74 bits, PositionTimeTime (54): add 85 bits

PositionTime	SEQUENCE			
Position				54 bits
Time				11 bits

<u>Summary: Uplink Message Size</u>

Message Variable	**Header Size**	**Variable Size**	**Total (octets)**
NULL	7 octets, 2 bits	0	8
Level (singleLevel)	7 octets, 2bits	1 octets, 3 bits	9
Level (blockLevel)	7 octets, 2bits	2 octets, 5 bits	10
LevelLevel	7 octets, 2bits	2 octets, 4 bits	10
Time	7 octets, 2bits	1 octets, 3 bits	9
Position	7 octets, 2bits	6 octets, 6 bits	14
PositionTime	7 octets, 2 bits	8 octets, 1 bit	16
PositionTimeTime	7 octets, 2 bits	10 octets, 5 bits	18
TimeLevel or LevelTime	7 octets, 2bits	2 octets, 6 bits	10
PositionLevel or LevelPosition	7 octets, 2bits	8 octets, 2 bits	16
PositionLevelLevel	7 octets, 2 bits	9 octets, 2 bits	17

4.8.1.6 *CPDLC AircraftPDUs [send]*

Common header (6 octets)

Element	***Type***	***Value***	***Encoding***	***Comments***
GroundPDUs	CHOICE	send	0 11	No extension used The CHOICE [3] is selected
ATCDownlinkMessage	SEQUENCE			
Header	SEQUENCE		00	Default value ’notRequired’ is selected
MessageIdNumber	INTEGER (0..63)	0	000.0000	
DateTime	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1997	0000.001	1997-1996
Month	INTEGER (1..12)	11	1010	11-1
Day	INTEGER (1..31)	2	0.0001	2-1
Timehhmmss	SEQUENCE			
Time	SEQUENCE			
Hours	INTEGER (0…23)	0	0000.0	
Minutes	INTEGER (0…59)	0	000000	
Seconds	INTEGER (0..59)	0	0.00000	
ElementsIds			001.	1 element id

ATCUplinkMsgElementId	CHOICE		0	No extension used
			0000000	CHOICE [msg number]

5. FIS APPLICATION

5.1 Introduction

5.1.1 Purpose

5.1.1.1 In line with normal ICAO practice, this chapter was developed as part of a companion document to the Flight Information Services (FIS) Standards and Recommended Practices (SARPs) [1]. It may be read alongside the FIS SARPs, in order to provide a greater understanding of the specification itself, or it may be read instead of the ATN FIS SARPs by readers that simply want to understand the purpose of the FIS Application rather than the detail of the specification.

5.1.1.2 This chapter also provides some historical information on the development of the FIS Application and explanations as to why the FIS Application is specified in the SARPs the way that it is, including further explanation of notes and recommendations.

5.1.2 Scope

5.1.2.1 This chapter provides guidance material for those implementing of the Automatic Terminal Information Service as part of the ATN Flight Information Service Application.

5.1.2.2 This chapter does not define any mandatory or optional requirements for the FIS Application, neither does it define any recommended practices. This chapter does not instruct users on how to use the FIS Application in a particular operational environment.

5.1.2.3 The FIS SARPs are dedicated to Air Traffic Services. Aeronautical Operational Control (AOC) may choose to use the FIS SARPs as a model for their own applications.

5.1.3 History

5.1.3.1 The FIS Application allows a pilot to request and receive FIS services from ground FIS systems. In a fully operational ATS data link environment, the FIS Application is expected to be used as the main means of passing flight information (e.g. automatic terminal information (ATIS), notices to airmen (NOTAMs), meteorological aerodrome reports (METARs) and extracts from Aeronautical Information Publications (AIPs)) to aircraft, whether in flight or on the ground.

5.1.3.2 In the initial implementation of FIS, only ATIS information will be passed. The FIS(ATIS) Application identifies the instance of the FIS Application dealing with ATIS information. ATIS messages, their format and intent, are based on the relevant ICAO documentation, in particular Annexes 3 and 11 and *Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services* (PANS-RAC, Doc 4444). The format and content of the messages will be identical to the current voice based systems.

5.1.3.3 The use of data link is not as flexible as voice, and a set of rules has had to be developed indicating, for example, how a dialogue is opened and closed, and how a particular sequence of messages within a dialogue is ended. However, the intention is that this should be as automatic as possible, with an apparently seamless line of communication between end users. The extent of automation will ultimately be the responsibility of the system designers, both from the engineering and operational aspects.

5.1.3.4 In addition to the ICAO documentation noted in paragraph 1.3.2 above, the main document from which the FIS SARPs have been developed is the ICAO *Manual of Air Traffic Services Data Link Applications* (Doc 9694) [3]. This specifies operating concepts in some detail. ICAO has specified that the FIS Application conforms to the ATN protocols for its data link operations.

5.1.3.5 The initial development of the SARPs centred around the requirement to replace a broadcast service with a service based on individual contact between the user (the aircraft) and the provider (the ground system). An ATIS broadcast system allows the pilot to obtain current information when required, and if the information becomes obtrusive, it can be switched off. If at any stage in the flight a controller detects that the aircraft is not in possession of the current information, it can easily be updated by voice.

5.1.3.6 This functionality is replicated to the extent possible by having two basic modes of operation in a data link FIS, namely a single request capability, and a 'contract' with the ground system, which provides updates to the aircraft as and when the information is updated by the ground. In stable conditions update rates may be virtually nil, whereas during the passage of an active front the ATIS may be updated several times per hour.

5.1.3.7 States might not be willing to incur the costs of implementing a complete system if they only ever intended to use certain elements of the application, e.g. although they would be obliged to make data link FIS information available on request, they may not wish to implement the provision of the update functionality. The SARPs therefore took account of the need to enable partial implementation whilst still retaining the interoperability required by the ICAO standards. This led to the development of subsetting rules, and the identification of conformant configurations.

5.1.3.8 The ATNP worked very closely with the ADSP, to ensure that the development of both the operational concepts, and the technical means of achieving them keep in step with each other. However, the ADSP looked generally at a longer timescale than the current ATNP initial implementation programme, and this inevitably meant that some elements of their work has not been incorporated into the present SARPs.

5.1.3.9 The ATNP identified a set of packages to accommodate the continued specification of operational requirements by the ADSP. This chapter provides guidance material for version 1 of the FIS Application contained in the first set of ATN SARPs (known as CNS/ATM-1 Package).

5.1.4 Structure

5.1.4.1 Section 5.1 — INTRODUCTION — contains the reason for providing guidance material as well as the scope. In addition, it provides a brief overview of the functionality of the FIS Application, FIS' relationship with other SARPs, and identifies applicable reference documents.

5.1.4.2 Section 5.2 — OVERALL GENERAL FUNCTIONALITY — describes generic concepts that are used throughout the FIS SARPs and guidance material. This chapter also covers some implementation issues that are not addressed in the FIS SARPs.

5.1.4.3 Section 5.3 — FIS SERVICE DESCRIPTION — gives a functional breakdown of the various services that FIS Application provides. It describes a peer to peer interaction, including reasons for why particular information is used or not used, and what actions on the information are expected.

5.1.4.4 Section 5.4 — FIS SARPs CHAPTER DESCRIPTION — clarifies any functionality that was not addressed in section 5.3 on a chapter by chapter basis.

5.1.4.5 Section 5.5 — DIMENSIONS — gives some sample encoding sizes for guidance on what capacities need to be allowed for in order to meet normal operational expectations.

5.1.4.6 Section 5.6 — ASN.1 INDEXES and TABLES — describes each ASN.1 field composing the FIS protocol data units as well as provides a cross-reference between fields.

5.1.4.7 Section 5.7 — EXAMPLE SCENARIOS — gives some examples as to what typical scenarios one can expect in course of normal FIS operation.

5.1.4.8 Section 5.8 — EXAMPLE ENCODING — outlines some actual sample PER encoding of typical FIS messages.

5.1.4.9 Section 5.9 — IDENTIFIED PROBLEMS — lists the known defects in the FIS SARPs considered are not jeopardizing the interoperability capability nor the safety of the FIS Application. These problems will be corrected in the next version of the FIS Application.

5.1.5 FIS Data Link Application Overview

5.1.5.1 *Summary*

5.1.5.1.1 The FIS Application described in the FIS SARPs allows a pilot to request and receive ATIS information from ground FIS systems via data link. It provides both air and ground users with the FIS Data Link Service limited to the ATIS information. The ATIS data link service supplements the existing availability of ATIS as a voice broadcast service, provided at aerodromes world-wide. All types of ATIS currently in use are encompassed (i.e. arrival, departure and combined).

5.1.5.1.2 The aircraft (pilot and/or avionics) requests the service by generating a request message for transmission to a FIS ground system. A FIS contract is then established by the FIS service provider which could take one of the two following forms:

a) the *FIS Demand Contract* where the ground FIS system provides the information once only; and

b) the *FIS Update Contract* where the ground FIS system provides the information and any subsequent update of this information.

5.1.5.1.3 These two types of FIS contract have been identified based on the analysis of the ATIS and METAR services as described in [3]. It is likely that additional types of contracts (e.g. FIS Periodic Contract) will be identified to support other data link FIS Services.

5.1.5.2 *Establishment and Operation of a FIS Demand Contract for ATIS Information*

5.1.5.2.1 This function allows the air system to establish a FIS demand contract with a FIS ground system, and then for the conditions of that contract to be realised. Realisation of the demand contract involves the sending of a single FIS report from the ground system to the aircraft, optionally after the sending of a positive acknowledgement.

5.1.5.2.2 Multiple FIS demand contracts may be established in parallel with the same ground FIS system.

5.1.5.2.3 The aircraft sends a FIS demand contract request to the FIS ground system. This contains an indication of which airport the requested ATIS information is related to. If the aircraft wants to retrieve the ATIS of several airports, it will establish one FIS demand contract per airport.

5.1.5.2.4 The ground FIS system then determines whether or not it is able to comply with the request:

a) if the ground FIS system detects that the requested ATIS information can be retrieved but is not yet available, the ground FIS system formats and sends to the airborne FIS system a FIS-positive-acknowledgement message first to indicate its acceptance of the contract, and the FIS-report message later;

b) if the ground FIS system can comply promptly with the FIS demand contract request it sends the FIS-report message as soon as possible; and

c) if there are errors in the FIS-demand-contract message, or if the ground FIS system cannot comply with the request, it sends a FIS-contract-reject message to the airborne FIS system indicating the reason for its inability to accept the contract.

5.1.5.3 *Establishment and Operation of a FIS Update Contract for ATIS Information*

5.1.5.3.1 This function allows the airborne FIS system to establish an Update Contract with a ground FIS system. Realisation of the update contract involves the sending of FIS reports from the ground FIS system to the aircraft each time the requested FIS information is modified.

5.1.5.3.2 Multiple FIS update contracts may be established in parallel with the same ground FIS system.

5.1.5.3.3 The airborne FIS system sends a FIS update contract request to the ground FIS system. This contains which airport the requested ATIS information is related to.

5.1.5.3.4 The ground FIS system then determines whether or not it is able to comply with the request:

a) if the ground FIS system can comply with the FIS-update-contract request, it sends the first FIS report as soon as possible, and whenever the requested FIS information is modified, it sends a new FIS report to the aircraft;

b) if the ground FIS system detects that the requested FIS information can be retrieved but is not yet available, then it sends to the airborne FIS system a FIS positive acknowledgement first to indicate its acceptance of the contract, and then sends the FIS reports;

c) if there are errors in the FIS update contract request, or if the ground FIS system cannot comply with the request, it sends a FIS-contract-reject message to the airborne FIS system indicating the reason for its inability to accept the contract; and

d) if the ground FIS system does not support the update contract function, it sends a FIS-contract-reject message containing, if available, the requested FIS information [1]

5.1.5.4 ***Cancellation of FIS Contracts***

5.1.5.4.1 This function allows both air and ground FIS systems to cancel a particular FIS update contract that is in operation.

5.1.5.4.2 A FIS-update-contract-cancel request is sent by the system initiating the termination. Any FIS report previously sent is delivered to the aircraft before the contract is effectively ended. Other pending FIS contracts are not disrupted by the termination of a particular FIS update contract.

5.1.5.4.3 The cancellation of the FIS update contract is confirmed to the FIS-user by the system receiving the FIS update contract cancel message.

5.1.5.4.4 The airborne FIS system may also cancel all FIS contracts (demand and update contracts) of the same type by sending a FIS-cancel-contracts request. The ground FIS system cancels these contracts and acknowledges the cancellation by sending a FIS-cancel-contracts-accept request. The cancellation is made on the basis of the type(s) of contract supplied by the airborne FIS system.

[1] Version 1 of the FIS Application does not allow the ground FIS server to send first a positive acknowledgment to the update contract request and then later a rejection of the update contract as specified in [3]. An ulterior version of the application will provide this capability.

5.1.5.5 ***Aborts***

5.1.5.5.1 This function allows the airborne system, the ground system or the communications system to abort a connection in cases where a serious problem has occurred.

5.1.5.5.2 If the communications part of the airborne or ground systems, or the network itself, detects an error, either in itself or in the protocol arriving from its peer, it will initiate a FIS provider abort.

5.1.5.5.3 If the user part of the airborne or the ground system detects an error, it has the option of initiating a FIS user abort.

5.1.5.5.4 In either of these cases, the result is that the connection is closed down immediately. Some of the messages already transmitted, but not yet confirmed, may be lost in transit.

5.1.6 Inter-relationships with Other SARPs

5.1.6.1 There is no interaction between the FIS SARPs and the other ATN Applications SARPs.

5.1.6.2 The FIS SARPs make use of the Upper Layer Application SARPs [2] to perform dialogue service functions required by the FIS Application.

5.1.7 Structure of the FIS SARPs

5.1.7.1 All the air-ground SARPs are produced to a standard format. This has greatly helped the maintenance of document stability, commonality and presentation. The FIS SARPs are no different in basic layout from all other air-ground applications SARPs.

5.1.7.2 The FIS SARPs constitute the fourth part of sub-volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705 — AN/956).

5.1.7.3 SARPs section 2.4.1 — INTRODUCTION — gives a very brief, high level description of FIS, as an application enabling FIS services to be provided to a pilot via the exchange of messages between aircraft avionics and ground FIS systems. Since this overview contains no information directly related to the stipulation of specific standards, it is almost entirely written as series of informative notes.

5.1.7.4 SARPs section 2.4.2 — GENERAL REQUIREMENTS — contains information and high level requirements for error processing and FIS version number requirements..

5.1.7.5 SARPs section 2.4.3 — ABSTRACT SERVICE — defines the abstract service interface for the FIS Application. The FIS Application Service Element (FIS ASE) abstract service is described from the viewpoint of the FIS-air-user, the FIS-ground-user and the FIS-service-provider.

5.1.7.6 SARPs section 2.4.4 — FORMAL DEFINITION OF MESSAGES — describes the contents of all permissible FIS messages through definition of the FIS ASN.1 abstract syntax. All possible combinations of message parameters and their range of values are detailed.

5.1.7.7 SARPs section 2.4.5 — PROTOCOL DEFINITION — splits up the specification of the FIS protocol into three parts: sequence diagrams for the services covered by the abstract service, protocol descriptions and error handling for the FIS-Air- and Ground-ASEs, and State Tables.

5.1.7.8 SARPs section 2.4.6 — COMMUNICATION REQUIREMENTS — specifies the use of Packed Encoding Rules (PER) to encode and decode the ASN.1 message structure and stipulates the Dialogue Service requirements, including Quality of Service (QoS).

5.1.7.9 SARPs section 2.4.7 — FIS USER REQUIREMENTS — describes the requirements imposed on the FIS-users concerning FIS messages and interfacing with the FIS ASEs.

5.1.7.10 SARPs section 2.4.8 — SUBSETTING RULES — specifies conformance requirements which all implementations of the FIS protocol obey. The protocol options are tabulated, and indication is given as to whether mandatory, optional or conditional support is required to ensure conformance to the SARPs. These subsetting rules will permit applications to be tailored to suit individual implementations, commensurate with the underlying task, while still maintaining an acceptable level of interoperability.

5.1.8 References

[1] Flight Information Services Application, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume II of the *Manual of Technical Provisions for the ATN* (Doc 9705), Section 2.4

[2] Upper Layer Communications Service, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume IV of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[3] *Manual of Air Traffic Services Data Link Applications* (Doc 9694)

[4] Introduction and System Level Requirements, Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume I of the *Manual of Technical Provisions for the ATN* (Doc 9705)

5.2 Overall General Functionality

5.2.1 General

5.2.1.1 Flight Information Services encompass several types of ground information as for instance meteorological data and aerodrome related data. For those data which are expected to be available via data link, the specification of the associated data link service (DFIS service) is required. The ADS Panel has produced the description of two DFIS services: ATIS service and METAR service.

5.2.1.2 The ATN application designed to support the DFIS services is the FIS Application. However, there is not just one FIS Application but several FIS(DFIS Service) Applications. There are as many FIS Applications as there are DFIS services. The FIS SARPs describe one of these applications: the FIS(ATIS) Application[2].

5.2.1.3 The FIS(ATIS) Application must be therefore considered as the first in a suite of air-ground applications meeting the whole range of requirements for Flight Information Services (FIS).

5.2.2 Air and Ground Topology

5.2.2.1 The FIS Application functions on a client/server architecture. FIS ground systems are servers providing FIS services to any clients, which are the data link equipped aircraft hosting the FIS application.

5.2.2.2 A FIS ground system maintains FIS information related to one or several DFIS services. For instance, a FIS ground system may be specialized for providing ATIS information only. This server is identified in Figure 5.2-1 as the FIS(ATIS) server number 1. On the other hand, a FIS ground system could provide several DFIS services, for instance both ATIS and METAR information. This server is illustrated in Figure 5.2-1 as the FIS(ATIS & METAR) server 2. There is no constraint in the FIS SARPs on how the FIS service providers must configure their FIS ground systems. The main advantage of the multi-service FIS ground system is that a unique connection established with the aircraft is used to support all FIS contracts.

5.2.2.3 Likewise, the airborne architecture for the FIS systems is not specified in the FIS SARPs. Aircraft manufacturers can choose to implement one FIS application entity per DFIS service offered to the aircrew or a single FIS application entity offering all the DFIS services. Aircraft 1 and 2 in Figure 5.2-1 are both equipped to provide ATIS and METAR but for aircraft 1 there are 2 distinct applications in operation while in aircraft 2 there is only one.

5.2.2.4 The selection of a particular ground topology does not impact the operational procedures for FIS. The FIS protocol is designed to allow the airborne FIS systems to inter-operate with a mono-service and a multi-service FIS ground system. If the requested service is not provided by the contact FIS ground system, the request is rejected with reason “service non available”.

5.2.2.5 The flexibility of the ground topology implies to assign an application name and an ATN address per DFIS service. When ATIS and METAR are provided through two distinct FIS ground systems, two addresses must be uplinked to the aircraft. In case both services are

[2] In this chapter, “FIS Application” is a generic term referring to all FIS(XXX) Data Link Applications whereas “FIS(ATIS) Application” refers to the specific instance of the FIS Application developed for providing the ATIS service.

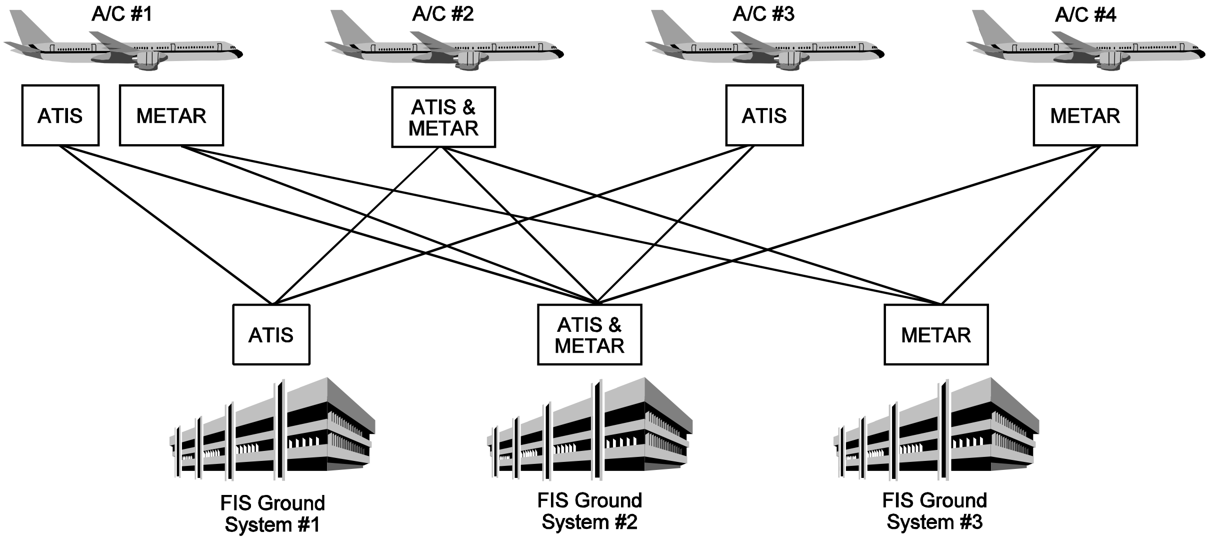


Figure 5.2-1. Possible configurations of the air and ground FIS Application Entities

provided by the same FIS ground system, the same ATN address is uplinked twice. This is the reason why [2] defines an AE-Qualifier value to ATIS (“ ATI”) and not to FIS.

5.2.2.6 Note that as version 1 of the FIS protocol only covers ATIS information, SARPs-compliant implementations are illustrated in Figure 5.2-1 by aircraft 3 and FIS ground system 1.

5.2.3 Airport coverage of the ATIS ground systems

5.2.3.1 A FIS(ATIS) ground system maintains ATIS information related to one or several airports. The alternatives are illustrated in Figure 5.2-2. The FIS(ATIS) server 1 is implemented in a particular airport to exclusively provide ATIS for this airport. The centralized FIS(ATIS) servers 2 and 3 provide ATIS information for several airports.

5.2.3.2 This flexible architecture permits multiple implementation choices concerning the ground topology definition. This choice is made by the authorities providing the FIS information based on their own criteria: for example accessibility of the FIS information, existence of communication infrastructure, distribution of the FIS addressing access points.

5.2.3.3 The choice “distributed versus centralized” FIS server has the following consequences:

As the airborne system must know the ATN address of the contacted FIS ground systems, the number of CM exchanges before establishing a FIS contract is dependent on the type of these systems: a single CM exchange is needed for a FIS ground system centralized at the regional scale whereas several CM exchanges may be required when several FIS ground systems cover the region.

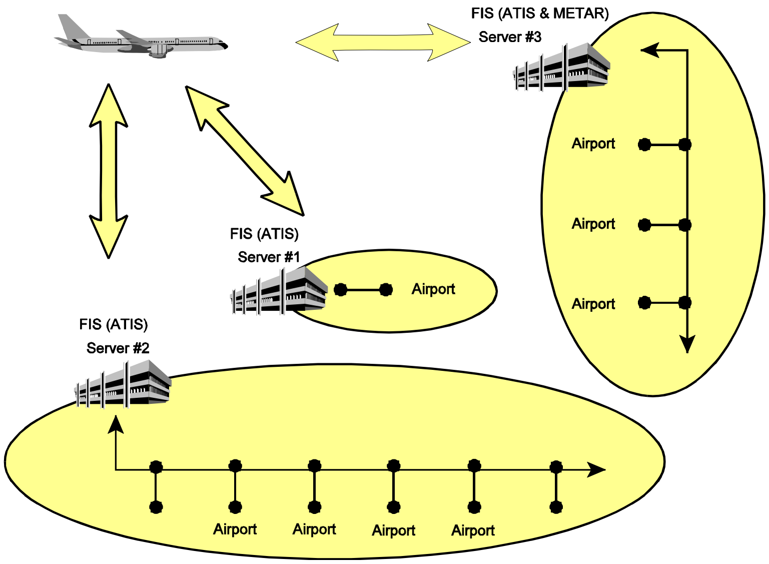


Figure 5.2-2. Centralized vs Distributed FIS servers

The centralized FIS server implies the availability of ground communications with the airports and the management in real-time of a centralized database.

5.2.3.4 Note that as version 1 of the FIS protocol only covers ATIS information, SARPs-compliant implementations are illustrated in Figure 5.2-2 by FIS ground systems 1 and 2.

5.2.3.5 How the information is conveyed and updated between the FIS ground systems and the actual sources of the FIS information (e.g. airports for ATIS) is not addressed by the SARPs. The ground exchanges are of the responsibility of the administrative authority producing the FIS information. They have to be defined locally or by bilateral agreements between authorities.

5.2.4 Abstract Internal Architecture

5.2.4.1 *General*

5.2.4.1.1 The architectural abstract model for the FIS Applications conforms to [2]. The abstract model instantiated for the mono-service (ATIS or METAR) and multi-service (ATIS and METAR) FIS Application is shown in Figure 5.2-3.

5.2.4.1.2 The main characteristics of this model are the following:

a) one application entity (AE) is defined per FIS Application;

b) each FIS AE contains the FIS Application Service Element (ASE) defined in the FIS SARPs. The FIS ASE is the communication element of the FIS Applications;

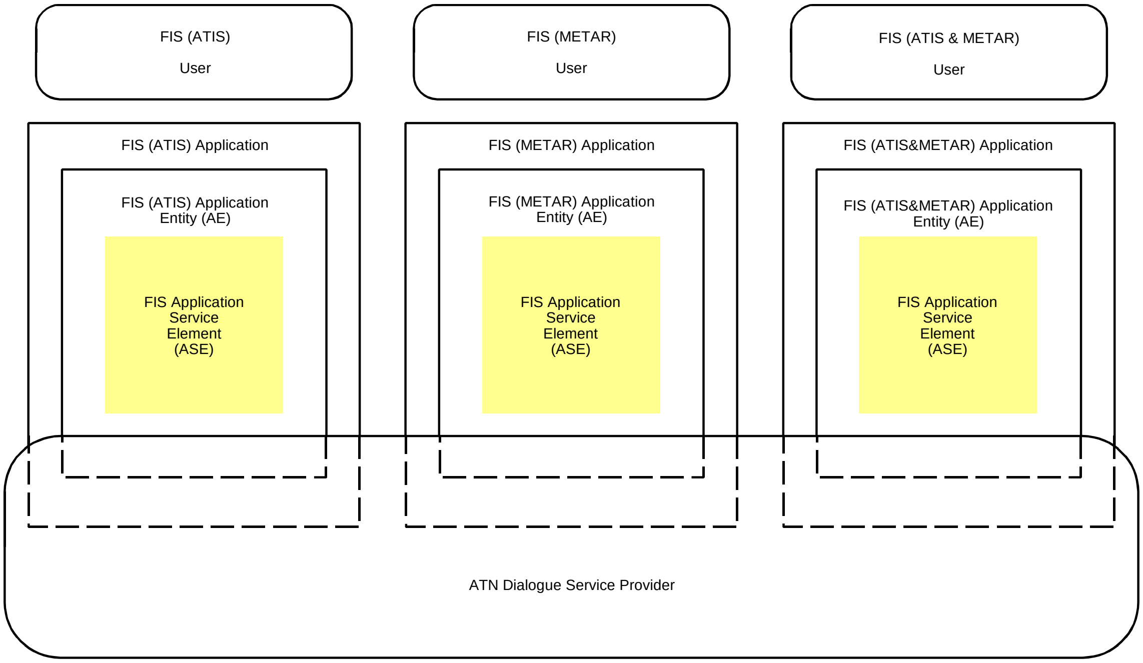

FIS (ATIS)
User
FIS (METAR)
User
FIS (ATIS & METAR)
User
FIS (ATIS) Application
FIS (METAR) Application
FIS (ATIS&METAR) Application
FIS (ATIS) Application Entity (AE)
FIS (METAR) Application Entity (AE)
FIS (ATIS&METAR) Application Entity (AE)
FIS Application Service Element (ASE)
FIS Application Service Element (ASE)
FIS Application Service Element (ASE)
ATN Dialogue Service Provider

Figure 5.2-3. Abstract Structure of the FIS Applications

c) the same ASE can be used to provide one or several DFIS services; and

d) the FIS ASE uses the Dialogue Service for communication with the peer ASE through the ATN.

5.2.4.1.3 Note that as version 1 of the FIS protocol only covers ATIS information, SARPs-compliant implementations will conform to the left-hand abstract structure illustrated in Figure 5.2-3.

5.2.4.2 *FIS ASE Functionality*

5.2.4.2.1 The FIS SARPs describe the FIS Application Service Element. The service provided by the FIS ASE at its top interface is provided as such by the FIS AE to the FIS-users.

5.2.4.2.2 The FIS ASE handles the FIS protocol. If individual functions are not supported as allowed by the subsetting rules, the ASE will ensure that the protocol will handle the remaining subset of the FIS functionality.

5.2.4.2.3 The FIS SARPs defines only one type of ASE – the FIS ASE for ATIS. The FIS ASE has two variants, the FIS-air-ASE and the FIS-ground-ASE.

5.2.4.3 *FIS-User Functionality*

5.2.4.3.1 In the model of FIS Application in Figure 5.2-3, there is a module called the FIS-user. The functionality of the FIS(ATIS)-user is described in SARPs section 2.4.7.

5.2.4.3.2 The FIS-air-user is responsible for initiating the FIS contracts and making the FIS-reports available to the end user, i.e. the pilot or the crew members.

5.2.4.3.3 The FIS-ground-user is responsible for the operation of the FIS contracts. On receipt of the contract request, it is responsible for responding to the request and then creating and submitting FIS reports in line with the contract.

5.2.4.4 *Internal Structure of the FIS ASE*

5.2.4.4.1 The internal abstract structure of the ATN ASEs is not standardised across applications.

5.2.4.4.2 The internal structure of the FIS ASE has been defined based on the following rationale: the procedures used to support the operation of the FIS contracts are common to most DFIS services. The major difference amongst FIS contracts is the structure of the operational data exchanged during the contract lifetime. It is therefore possible to define generic ASE modules common to several FIS Applications.

5.2.4.4.3 As far as the FIS Application is concerned, four generic FIS ASE modules have been defined for handling the ATIS protocol:

a) the FIS Demand Contract (DC) module,

b) the FIS Update Contract (UC) module,

c) the FIS Cancel Contract (CL) module, and

d) the FIS Abort Contract (AB) module.

5.2.4.4.4 The abstract internal structure of both air and ground FIS ASEs is illustrated in Figure 5.2-4. Two interface modules have been defined to cope with the interface with the ASE-users (High Interface (HI) module) and with the Dialogue Service Interface (Low Interface (LI) module).

5.2.4.4.5 This flexible architecture allows for the reuse of common FIS ASE modules in the FIS Applications that will be specified in the future. In addition, additional ASE modules could be added in the ASE to provide additional services.

5.2.4.4.6 Section 5.3 describes the data flows within the FIS ASE when a primitive is received by the interface modules.

5.2.4.5 *Product Architecture*

5.2.4.5.1 The SARPs have defined an abstract model for the purposes of definition. That is, it has split the functionality between the ASE and the user and has defined an abstract interface between the two. It is strongly emphasized that there is no requirement on implementors to build the interface defined in the SARPs between the ASE and the user. If it is

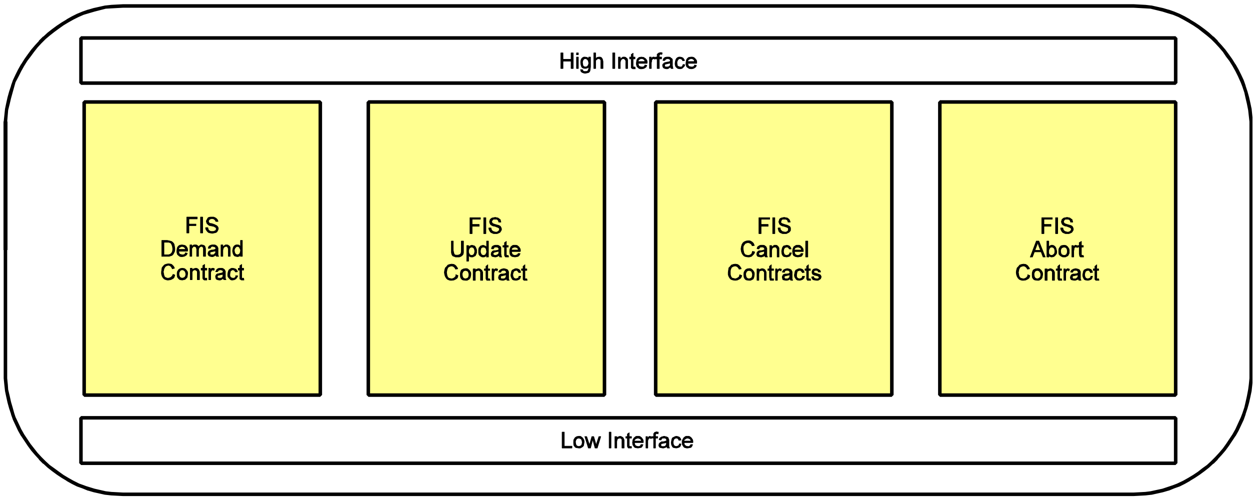


Figure 5.2-4. Abstract Structure of the FIS ASE

convenient, from an engineering perspective, to build an interface between two modules that embody the functionality of the ASE and the user, then the implementors are free to do so. However, if it is more convenient to build the system with interfaces in other places, then that is also acceptable. In testing a product to see if it conforms to the SARPs, no test can be made to test internal interfaces within the system.

5.2.4.5.2 The internal structure of the ATN ASE is not standardized across applications; for instance, FIS ASE is defined as several modules while CM is defined as a single module.

5.2.5 Implementation Dependent Functionality

5.2.5.1 The FIS SARPs specify some of the requirements for the user, but leave a lot up to the implementors. There are no requirements that state how the user interface appears, how FIS interacts with the systems generating the ATIS information, how FIS interacts with higher level functionality and with other applications such as CM. All this is implementation dependent.

5.2.5.2 The definition of the ATIS reports in the SARPs is quite open. It has not been possible to specify with the ASN.1 notation all the constraints and relationships between the ATIS fields. The rules for building a consistent ATIS report are not checked by the FIS ASEs and have to be implemented by the FIS-users.

5.2.6 Rationale for ASE / Users Split

5.2.6.1 The rationale for the split in functionality between the FIS ASEs and the FIS-users is as follows:

5.2.6.1.1 The ASE contains all functionality that is necessary to ensure the interoperability at the syntactic level. That is, two valid implementations of ASEs will be able to interact, passing data to each other in the correct order; they will be able to check the format of the data, ensure that it has been sent with the appropriate dialogue service primitive and also ensure that the peer ASE is behaving according to the requirements in the SARPs. The ASE thus ensures interoperability.

5.2.6.1.2 SARPs Chapter 7 defines some requirements for the users. These are minimum user requirements necessary to ensure the semantic interoperability of the two peers. Thus it explains how each FIS contract is interpreted and how it must operate.

5.2.6.1.3 Some care has been taken to ensure that the requirements are not over-specified. That is, they do not specify rules which are not absolutely essential to the syntactic and semantic interoperability of the FIS function. This implementation dependent part can be built by different manufacturers in different ways, without affecting the interoperability between different implementations. This implementation dependent part has not been specified in the SARPs, but must be specified by individual product manufacturers or regional standards.

5.2.7 Inter-relationship with other ATN Applications

5.2.7.1 There is no direct interaction required between the FIS Application and the other ATN applications. The FIS Application is a stand-alone application which can be developed, certified, installed and operated completely independently from the other ATN Applications.

5.2.7.2 However, in order to be able to initiate a FIS dialogue, some naming and addressing information have to be known from the dialogue initiator:

a) the ATN address of the ground FIS system must be present somewhere in the data link communication system. Without this information, the dialogue with the ground FIS system cannot be established; and

b) the version number of the FIS protocol run on the ground FIS system must be available to the FIS-air-user. The protocol version negotiation is not performed by the FIS Application Entities and the FIS-air-user is responsible for checking the version compatibility before establishing a FIS Contract.

5.2.7.3 The Context Management (CM) Application is the most natural way to exchange this addressing information. Other solutions could be chosen, like the hard-coding of the addressing data bases or the loading of this information at the gate before the take-off. When CM is used, a CM exchange is required between the aircraft and the ground CM bsystem administrating the contacted FIS system. There is no direct interaction between the CM Application and the FIS Application (in fact they are not running at the same time), but the addressing information exchanged by the CM Application is required to establish a FIS contract.

5.2.7.4 The definition of the ground topology for CM must take into account the presence of the FIS ground systems as follows:

a) a CM area cannot contain more than one FIS ground system delivering the same DFIS service of a same version number, and

b) a FIS ground system can belong to more than one CM area.

5.2.7.5 Figure 5.2-5 illustrates how distributed and centralized FIS servers can be assigned to CM areas.

5.2.7.6 When the pilot requests the ATIS of a particular airport, the identification and the address of a CM ground system attached to the ATIS server covering this airport must be known in the aircraft. A CM Logon must be performed with this ground system in order to retrieve the address of the ATIS server.

5.2.8 Ground FIS Exchanges

5.2.8.1 No operational requirement related to the ground FIS exchanges have been specified in [3]. As a consequence, there is no ground FIS protocol defined between FIS-ground-users in version 1 of the FIS Application.

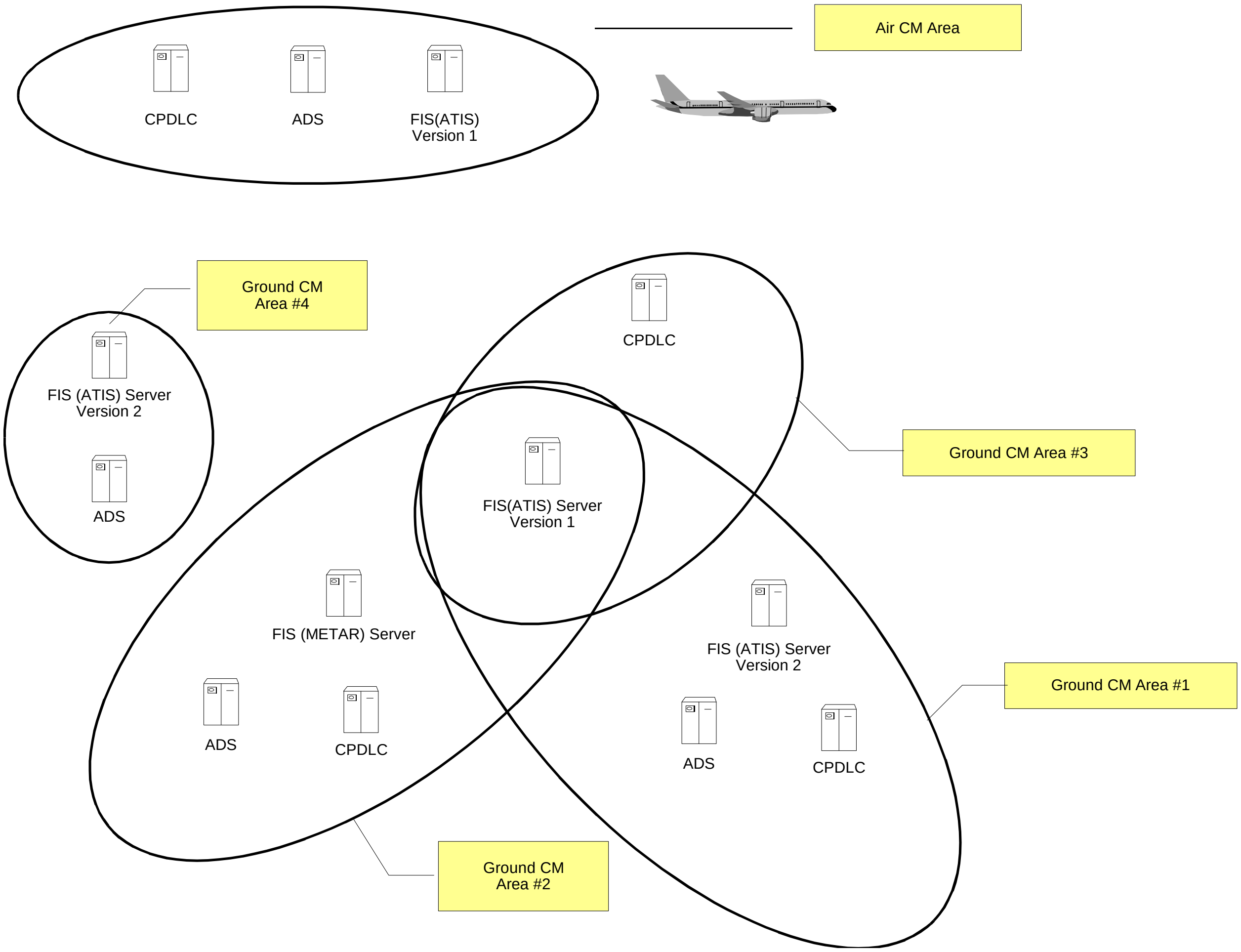


Figure 5.2-5. Example of ground FIS topology when CM is used

5.2.8.2 As a consequence of the non availability of ground communications between FIS ground systems, a request sent by an aircraft to a FIS ground system cannot be forwarded to another FIS ground system. Thus, FIS requests must be sent in the first place to the appropriate ground FIS system, i.e. the one which holds the requested information. If the requested information is not maintained by the FIS ground system receiving the request, the request is rejected.

5.2.8.3 FIS ground forwarding exchanges may be considered in the scope of version 2 of the FIS Application. In the meantime, ground FIS exchanges may be implemented by some states (based on bilateral agreements) between FIS ground systems in order to allow the forwarding of both FIS requests and reports. This topology is outside the scope of the FIS SARPs.

5.2.9 Dialogue Management

5.2.9.1 *General*

5.2.9.1.1 The term “dialogue” refers here the end-to-end communication path provided by the Dialogue Service Provider. The Dialogue Service is described in detail in [2]. A dialogue is mapped directly onto a transport connection.

5.2.9.1.2 The dialogues supporting the FIS contracts are always initiated by the FIS airborne system. In like manner, the FIS airborne system is always responsible for closing the dialogue.

5.2.9.1.3 The dialogue service is used by the FIS ASEs for the following purposes:

a) establishment, graceful release and abort of a dialogue;

b) transfer of data;

c) support for quality of service; and

d) application naming.

5.2.9.2 *Optimisation of the use of dialogues*

5.2.9.2.1 The FIS service hides the use of the underlying communication service to the FIS-users: the FIS-users are aware when FIS contracts are established and cancelled. However, they are not necessarily aware of when the dialogues are started and ended.

5.2.9.2.2 Multiplexing over a single dialogue of FIS contracts set up between an aircraft and a ground FIS system is supported by the FIS protocol. Up to 256 FIS contracts can be multiplexed in parallel on a single dialogue. The time required to establish the dialogue before any operational data can be exchanged penalises only the first FIS contract. The dialogue multiplexing is performed by the Low Interface Module.

5.2.9.3 *Dialogue Establishment*

5.2.9.3.1 When the establishment of a new FIS contract is requested by the FIS-air-user, if no dialogue was established with the ground FIS system (i.e. if no connection has already been established), a dialogue is established first, then data related to the FIS contract are then exchanged. During the time of the dialogue establishment, no new FIS service request is accepted by the FIS-service provider.

5.2.9.3.2 On receipt of a FIS contract establishment request, if a dialogue is still in place and was maintained for a future use, it is used immediately for data transmission.

5.2.9.3.3 The SARPs prohibit an FIS-ground-ASE from requesting the establishment of a dialogue.

5.2.9.4 *Dialogue Release*

5.2.9.4.1 The dialogue is closed when no activity takes place during a certain period of time. Without explicit action from the pilot or a ground operator, the air FIS system triggers the release of the dialogue in place. The value for the inactivity timer is not standardised in the FIS SARPs. It could be customised for each airborne system.

5.2.9.4.2 The dialogue can be abruptly terminated by an abort condition at any time after the D-START request has been issued by the FIS-air-ASE.

5.2.9.4.3 The SARPs prohibit an FIS-ground-ASE from requesting the release of a dialogue.

5.2.10 **Protocol Monitoring**

5.2.10.1 *General*

5.2.10.1.1 The FIS ASE controls that the protocol is correctly handled by the peer and can be correctly operated locally. The generation and transmission of expected responses are monitored by the peer FIS ASE.

5.2.10.1.2 In case a FIS service provider timer expires or an exception is detected, the dialogue in place with the peer FIS system is aborted and all contracts in place or being established are cancelled. Active FIS-users are informed of this situation by a FIS-provider-abort indication and are provided with a reason for the abort. These cases are described in the following sections.

5.2.10.2 *FIS Service Provider Timers*

5.2.10.2.1 If the APDU corresponding to an FIS-demand-contract, FIS-update-contract, FIS-cancel-update-contract or FIS-cancel-contracts confirmation or a FIS-report indication is not received by the requesting FIS ASE within a locally specified period of time (SARPs section 2.4.5.2 recommends 6 minutes), the dialogue being established or already established is aborted. Both FIS-users are informed of this situation by a FIS-provider-abort indication with a reason “timer expiration”. This timer is a FIS service provider timer and is not directly connected to any operational timers set, except that it is sufficiently larger than any operational timer to prevent "nuisance" timer-expiration aborts.

5.2.10.2.2 Note that the expiration of the "t_inactivity" timer is considered as a normal event and does not raise an exception. It means that there has been no traffic on the maintained dialogue with the ground ASE for a period of time. In that case, the dialogue must be shut down by the FIS-air-ASE by using the confirmed dialogue service D-END.

5.2.10.2.3 The reception of the D-END confirmation is also monitored. If not received, the ultimate action is for the FIS-air ASE to try to abort the dialogue. As the FIS-users are not active any more, no FIS-abort indication is generated.

5.2.10.3 *Exception Handling*

5.2.10.3.1 **The expected APDU is missing**

5.2.10.3.1.1 With the exception of the D-END primitives, if the user data parameter of a received indication or confirmation does not contain an APDU, this means that the peer has not run correctly the FIS protocol. The dialogue is aborted by the ASE receiving the empty primitive with the reason "protocol error".

5.2.10.3.2 **The received primitive or APDU was not expected**

5.2.10.3.2.1 On receipt of a dialogue service indication or confirmation, the ASE checks that the APDU transmitted in the user data parameter of this primitive is authorised in its current state. If the received APDU is not authorised, this means that the peer made an error and does not run the protocol correctly anymore. The dialogue is therefore aborted by the ASE receiving the APDU with the reason "sequence error".

5.2.10.3.2.2 On receipt of a dialogue service indication or confirmation, the ASE checks that the invoked dialogue service is authorised in its current state. If the primitive is not authorised, this means that the sequencing of the dialogue primitives defined for the FIS protocol has not been respected by the peer. The dialogue is therefore aborted by the ASE receiving the out of sequence primitive with the reason "sequence error".

5.2.10.3.3 The received APDU cannot be decoded

5.2.10.3.3.1 If the received PDU cannot be decoded, a decoding error is raised highlighting a transmission error or an encoding error by the APDU sender. The dialogue is therefore aborted by the ASE receiving the invalid APDU with the reason "decoding error".

5.2.10.3.4 An Unrecoverable Internal Error occurred

5.2.10.3.4.1 The unrecoverable system error is intended to cover cases where a fault causes a system lockup or the system to become unstable (e.g. the system get short of memory). If the system has enough resources to do it, it will abort the connection and inform both the local user and the peer of the situation with the abort reason "unrecoverable internal error".

5.2.10.3.4.2 The unrecoverable system error is written as a recommendation in the SARPs instead of a requirement, as it is recognised that depending on the nature of the error in the system, it may not be possible to regain control in order to either perform an abort, or inform the user of the abort situation.

5.2.10.3.5 An Invalid Contract Number has been used

5.2.10.3.5.1 The contract number supplied by the FIS-users must be unallocated when establishing a new contract. Otherwise the contract number must correspond to an existing contract. If these rules are not respected, the ASE detecting the error aborts the dialogue with the abort reason "invalid contract number".

5.2.10.3.6 The End dialogue has been rejected by the peer

5.2.10.3.6.1 When the t_inactivity timer expires, the D-END request is invoked to shut down the dialogue. Upon receipt of the corresponding D-END indication, the FIS-ground-ASE's only choice is to accept the D-END by setting the D-END response Result parameter to the abstract value "accepted". If for some reason this value is not present (as checked in SARPs section 2.4.5.3.12.13), then the FIS-air-ASE will abort the dialogue with reason "dialogue-end-not-accepted". Since there are no active users any more, no indication is given to the FIS-users.

5.2.10.3.7 The requested QOS does not match the one specified in the FIS SARPs

5.2.10.3.7.1 The SARPs Sub-Volume I [4] dictates the values used for application service priority and RER quality of service parameters for all ATN applications. These values must be adhered to so proper levels of flight safety and performance are maintained. For FIS(ATIS), the values used are "aeronautical information service messages" for application service priority and "low" for RER quality of service. If these values are not present (as checked in SARPs section 2.4.5.4.6.1), then the FIS ASE will abort with reason "invalid-QOS-parameter".

5.2.11 Version Number Negotiation

5.2.11.1 The FIS SARPs specify the operation of version 1 of the FIS Application. The version number is a value inherent to the FIS ASE and is not provided by the FIS-users.

5.2.11.2 The version numbers supported by the air and ground systems for the FIS application are exchanged during the CM procedures. The FIS-air-user must check whether the ASE implemented onboard is compatible with one of the versions implemented on the ground.

5.3 Functionality of Services

5.3.1 Introduction

5.3.1.1 This section describes first the information provided by the ASEs to the FIS-users or supplied by the FIS-users to the ASEs. Then, it provides a high-level overview of the data flow within the ASE expected during normal operation. The data flow illustrated in SARPs Figure 2.4.5-19 is reproduced in Figure 5.3-1.

5.3.1.2 The primitives are grouped according to the services they provide: setting a FIS Demand Contract, setting a FIS Update Contract, cancelling and aborting FIS contracts.

5.3.2 Concepts

5.3.2.1 Users of the FIS service are termed *FIS-ground-user* and *FIS-air-user*. When it applies to both, it is termed *FIS-user*. The FIS-user represents the operational part of the FIS system. It is either the final end-user (e.g. a crew member or controller) or a automated system. The FIS-air-user that initiates a FIS Contract is termed the *calling* FIS-user or *initiator*. The FIS-ground-user that the initiator is trying to contact is termed the *called* FIS-user or *responder*.

5.3.3 FIS Service Parameters

5.3.3.1 *FIS Contract Number*

5.3.3.1.1 The *FISContractNumber* parameter identifies unambiguously any contract in place between a peer of FIS-users.

5.3.3.1.2 This identifier is allocated by the FIS-air-user when establishing a new FIS contract and supplied to the FIS-air-ASE as the *FISContractNumber* parameter of a FIS-demand-contract request or a FIS-update-contract request primitive. It is then supplied by the FIS-users in any subsequent request or response primitives or provided by the ASE in any subsequent indication or confirmation primitives related to a specific FIS contract.

5.3.3.1.3 Unallocated number must be used when invoking a FIS-demand/update-contract request. The invocation of terminal primitives (such as FIS-demand/update-contract confirmation

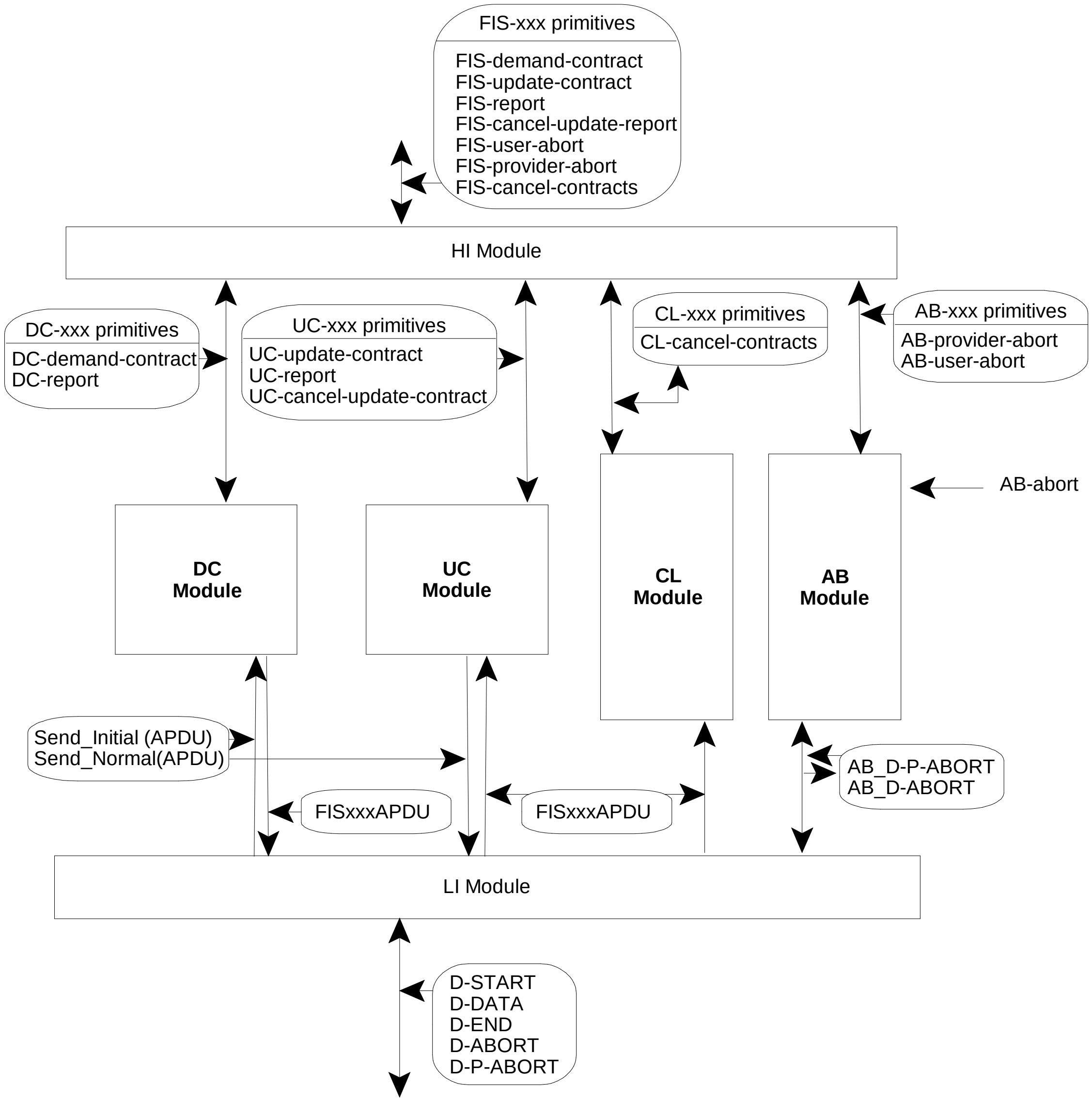


Figure 5.3-1. Data Flows in the FIS ASE

(accepted) or FIS-user-abort request/indication) causes the de-allocation of the current contract number(s) which could be used for a new FIS contract.

5.3.3.1.4 The FIS contract number supplied by the FIS-ground-ASE when invoking the FIS-report service or the FIS-cancel-update-contract service must have been assigned previously to an existing FIS contract.

5.3.3.1.5 The *FISContractNumber* parameter is an integer value in the range 1 to 256.

5.3.3.2 ***ICAO Facility Designation***

5.3.3.2.1 The *ICAOFacilityDesignation* parameter is used to indicate the facility designation of the called ground FIS system. It is used for dialogue service addressing purposes.

5.3.3.2.2 The *ICAOFacilityDesignation* parameter can be four to eight characters. Four characters are use to identify a particular facility, such as “ZYVR” for Vancouver. Up to eight characters may be used to identify a particular system within a facility; i.e. “ZYVRA123" may be the address for Vancouver en-route.

5.3.3.3 ***Class Of Communication Service***

5.3.3.3.1 The services FIS-demand-contract and FIS-update-contract each has an optional parameter called “*ClassOfCommunicationService*”. The *ClassofCommunicationService* parameter is a means for the FIS-air-user to indicate the required performance in terms of end-to-end transit delay. Values for these transit delays are given in Sub-Volume 1 of the ATN SARPs (section 1.3.7, Note 2).

5.3.3.3.2 The *ClassOfCommunicationService* parameter, if provided by the FIS-air-user, is supplied to the Transport Service Provider (TSP) in the T-CONNECT *SecurityLabel* parameter when the connection is established.

5.3.3.3.3 The Class of Communication Service is not guaranteed nor is a degradation of the provided class indicated to the users. It is of the responsibility of the application users to determine the actual transit delay achieved by local means such as message time stamping.

5.3.3.3.4 There is no negotiation of the Class of Communication Service between air and ground FIS-users. The value of the Class of Communication requested by an FIS-air-user for a dialogue is not transmitted nor indicated to the FIS-ground-user.

5.3.3.3.5 If the FIS-air-user does not require a particular Class of Communication, the *ClassofCommunicationService* parameter may be left blank indicating no routing preference. That means that the Class of Communication Service is chosen by the Dialogue Service Provider (“ ATSC: No Traffic Type Policy Preference”).

5.3.3.3.6 The *ClassOfCommunicationService* parameter is also used to implicitly indicate that the Routing Class is ATSC.

5.3.3.4 ***FIS Contract Details***

5.3.3.4.1 The *FISContractDetails* parameter contains the details of the FIS Contract requested by the FIS-air-user (i.e. the contract number, the contract type, the FIS service type, the airport and the ATIS type requested). In version 1 of the FIS protocol only ATIS information may be requested.

5.3.3.4.2 The *FISContractDetails* parameter is supplied by the FIS-air-user when requesting the establishment of a FIS demand contract or a FIS update contract.

5.3.3.4.3 The *FISContractDetails* parameter is passed transparently to the FIS-ground-user in the corresponding indication. However, the FIS service provider checks before the data encoding that the values supplied by the user are compatible with the syntactical description of the data types.

5.3.3.5 *FIS Information*

5.3.3.5.1 The *FISInformation* parameter contains the FIS information as requested by the FIS-air-user. In version 1 of the FIS protocol, only ATIS data element may be included in this parameter: arrivalATIS, departureATIS, combined ATIS or both arrival and departure ATIS.

5.3.3.5.2 The *FISInformation* parameter is supplied by the FIS-ground-user when accepting a FIS contract and when sending an update of the requested FIS information. It may also optionally be used by an FIS-ground-user not supporting the update functionality when rejecting the FIS update contract.

5.3.3.5.3 The *FISInformation* parameter is passed transparently to the FIS-air-user. However, the FIS service provider checks before the data encoding that the values supplied by the user are compatible with the syntactical description of the data types.

5.3.3.6 *Result*

5.3.3.6.1 The *Result* parameter is used by the FIS-ground-user to indicate in a FIS-demand/update-contract response whether it accepts or rejects the FIS demand/update contract.

5.3.3.6.2 The *Result* parameter is passed transparently to the FIS-air-user in the corresponding FIS-demand/update-contract confirmation.

5.3.3.7 *Reject Reason*

5.3.3.7.1 The *RejectReason* parameter is used by the FIS-ground-user to indicate why it rejects the FIS-demand/update contract.

5.3.3.7.2 The abstract value “canNotComply” is used when the request is valid but the requested information is not available (e.g. the ATIS request for a particular airport is not sent to the appropriate FIS ground server).

5.3.3.7.3 The abstract value “FISServiceUnavailable” is used when the service requested is not supported by the FIS ground server (e.g. a METAR request received by a FIS ground server dedicated to ATIS).

5.3.3.7.4 The abstract value “erroInRequest” is used when the request can be decoded but the values are wrong.

5.3.3.7.5 The abstract value “undefined” is used for all other cases of rejection.

5.3.3.7.6 The *RejectReason* parameter can be provided by the FIS-ground-user only when it rejects the establishment of the FIS contract.

5.3.3.7.7 The *RejectReason* parameter is passed transparently to the FIS-air-user in the corresponding FIS-demand/update-contract confirmation.

5.3.3.8 ***Reason***

5.3.3.8.1 The *Reason* parameter contains the reason for an abort initiated by the FIS service provider. It is indicated to the users in the FIS-abort indication primitive.

5.3.3.8.2

The possible abstract values for the *Reason* parameter are described in section 7.5.

5.3.3.9 ***FIS Service Type***

5.3.3.9.1 The *FISServiceType* parameter is used by the FIS-users to indicate which type of FIS contracts has to be cancelled when a multiple contracts cancellation is requested. In version 1 of the FIS protocol, only ATIS contracts can be cancelled with the FIS-cancel-contracts services.

5.3.4 **Setting a FIS Demand Contract**

5.3.4.1 The *FIS-demand-contract* service is used to set up a FIS Demand Contract between a FIS-air-user and a FIS-ground-user. It is a confirmed service, initiated by the FIS-air-user.

a) the FIS-demand-contract request is passed to the air HI module which examines it to see which other module to pass it on to. The request is passed to the air DC module. The FIS Contract Number attached to the new demand contract is stored by the air HI module;

b) the air DC module generates a FISDownlinkAPDU [FISRequest] APDU. It passes the APDU to the air LI module with the ground system identifier and the class of communication supplied by the user. Timers t-DC-1 and t-DC-2 are started to monitor the reception of the reply from the ground ([FISAccept] APDU or [FISReject] APDU) and the reception of the FIS report when the response is postponed;

c) the air LI module decides how to use the dialogue service. If a dialogue exists already, it makes use of that dialogue and uses the D-DATA service to pass the APDU to the ground system. If no dialogue exists, it uses the D-START service to pass the APDU to the ground system. The APDU is passed to the ground system by way of the upper and lower layers and emerges in the ground LI module;

d) the ground LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the ground DC module;

e) the ground DC module recognises the APDU as FISDownlinkAPDU [FISRequest] APDU and passes a FIS-demand-contract indication to the ground HI module; and

f) the ground HI module passes it to the FIS-ground-user.

5.3.4.2 The *FIS-demand-contract* service may be either accepted (in that case, a FIS report is enclosed in the response message), rejected (a reject reason is provided) or postponed (the FIS report is sent later using the FIS-report service) by the ground FIS system.

a) the FIS-demand-contract response is passed to the ground HI module which examines it to see which other module to pass it on to. The request is passed to the ground DC module;

b) the ground DC module generates a FISUplinkAPDU [FISAccept] or [FISReject] APDU based on the reply supplied by the FIS-ground-user. It passes the APDU to the ground LI module;

c) the ground LI module then passes the APDU to the air using a D-DATA request (if a dialogue is already open) or a D-START response (if no D-START response has yet been given). The APDU is passed to the airborne system by way of the upper and lower layers and emerges in the air LI module;

d) the air LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the air DC module;

e) the air DC module recognises the APDU as FISUplinkAPDU [FISAccept] or [FISReject] APDU and accordingly passes a positive or negative FIS-demand-contract confirmation to the air HI module. If the response is not postponed and if there is no FIS contract in place, the timer t-inactivity is started; and

f) the air HI module passes it to the FIS-air-user.

5.3.4.3 The *FIS-report* service is then invoked by the FIS-ground-user to both send a FIS report and close a previously postponed FIS Demand Contract. It is an unconfirmed service initiated by the FIS-ground-user.

a) the FIS-report request is passed to the ground HI module which examines it to see which other module to pass it on to. The decision is taken based on the FIS contract number supplied by the user and the contracts already managed and identified by the FIS ASE. The request is passed to the ground DC module;

b) the ground DC module generates a FISUplinkAPDU [FISReport]. It passes the APDU to the ground LI module;

c) the ground LI module then passes the APDU to the air using a D-DATA request. The APDU is passed to the airborne system by way of the upper and lower layers and emerges in the air LI module;

d) the air LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the air DC module;

e) the air DC module recognises the APDU as FISUplinkAPDU [FISReport] APDU and passes a FIS-report indication to the air HI module. If there is no FIS contract in place, the timer t-inactivity is started; and

f) the air HI module passes it to the FIS-air-user.

5.3.5 Setting a FIS Update Contract

5.3.5.1 The *FIS-update-contract* service sets up a FIS Update Contract between a FIS-air-user and a FIS-ground-user. It is a confirmed service, initiated by the FIS-air-user.

a) the FIS-update-contract request is passed to the air HI module which examines it to see which other module to pass it on to. The request is passed to the air UC module. The FIS Contract Number attached to the new demand contract is stored by the air HI module;

b) the air UC module generates a FISDownlinkAPDU [FISRequest] APDU. It passes the APDU to the air LI module with the ground system identifier and the class of communication supplied by the user. Timers t-UC-1 and t-UC-2 are started to monitor the reception of the reply from the ground ([FISAccept] APDU or [FISReject] APDU) and the reception of the FIS report when the response is postponed;

c) the air LI module decides how to use the dialogue service. If a dialogue exists already, it makes use of that dialogue and uses the D-DATA service to pass the APDU to the ground system. If no dialogue exists, it uses the D-START service to pass the APDU to the ground system. The APDU is passed to the ground system by way of the upper and lower layers and emerges in the ground LI module;

d) the ground LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the ground UC module;

e) the ground UC module recognises the APDU as FISDownlinkAPDU [FISRequest] APDU and passes a FIS-update-contract indication to the ground HI module; and

f) the ground HI module passes it to the FIS-ground-user.

5.3.5.2 The update contract may be either accepted (in that case, the FIS report is enclosed in the response message), rejected (a reject reason is provided, optionally the FIS report is enclosed in the response message) or postponed (the FIS report is sent later) by the ground FIS system. A FIS Update Contract must be explicitly closed by the FIS-users.

a) the FIS-update-contract response is passed to the ground HI module which examines it to see which other module to pass it on to. The request is passed to the ground UC module;

b) the ground UC module generates a FISUplinkAPDU [FISAccept] or [FISReject] APDU based on the reply supplied by the FIS-ground-user. It passes the APDU to the ground LI module;

c) the ground LI module then passes the APDU to the air using a D-DATA request (if a dialogue is already open) or a D-START response (if no D-START response has yet been given). The APDU is passed to the airborne system by way of the upper and lower layers and emerges in the air LI module;

d) the air LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the air UC module;

e) the air UC module recognises the APDU as FISUplinkAPDU [FISAccept] or [FISReject] APDU and accordingly passes a positive or negative FIS-update-contract confirmation to the air HI module; and

f) the air HI module passes it to the FIS-air-user.

5.3.5.3 The *FIS-report* service is then invoked by the FIS-ground-user to activate a postponed FIS Update Contract and to uplink any update of the FIS information. It is an unconfirmed service initiated by the FIS-ground-user. FIS reports are delivered to the FIS-air-user in the order they have been supplied by the FIS-ground-user.

a) the FIS-report request is passed to the ground HI module which examines it to see which other module to pass it on to. The decision is taken based on the FIS contract number supplied by the user and the contracts already managed and identified by the FIS ASE. The request is passed to the ground UC module;

b) the ground UC module generates a FISUplinkAPDU [FISReport]. It passes the APDU to the ground LI module;

c) the ground LI module then passes the APDU to the air using a D-DATA request. The APDU is passed to the airborne system by way of the upper and lower layers and emerges in the air LI module;

d) the air LI module examines the APDU in order to decide which module to pass it to based on the FIS contract number. The APDU is passed to the air UC module;

e) the air UC module recognises the APDU as FISUplinkAPDU [FISReport] APDU and passes a FIS-report indication to the air HI module; and

f) the air HI module passes it to the FIS-air-user.

5.3.6 Cancelling FIS Contracts

5.3.6.1 *Cancelling several contracts*

5.3.6.1.1 The *FIS-cancel-contracts* service allows the FIS-air-user to cancel all FIS demand and update contracts of the same type with a particular FIS-ground-user. This service does not affect the FIS contracts of other types still in place.

a) the FIS-cancel-contracts request is passed to the air HI module which examines it to see which other module to pass it on to. The request is passed to the air CL module;

b) the air CL module generates a FISDownlinkAPDU [FISCancelContracts] APDU. It requests the air DC and UC modules to stop operation. It passes the APDU to the air LI module. Timer t-CL-1 is started to monitor the reception of the reply from the ground ([FISCancelContractsAccept] APDU;

c) the air LI module then passes the APDU to the ground using a D-DATA request. The APDU is passed to the ground system by way of the upper and lower layers and emerges in the ground LI module;

d) the ground LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the ground CL module;

e) the ground CL module recognises the APDU as FISDownlinkAPDU [FISCancelContracts] APDU and passes a FIS-cancel-contracts indication to the ground HI module. It requests the ground DC and UC modules to stop operation; and

f) the ground HI module passes the FIS-cancel-contracts indication to the FIS-ground-user.

5.3.6.1.2 The *FIS-cancel-contracts* service is a confirmed service initiated by the FIS-air-user. As the cancellation cannot be rejected by the FIS-ground-user, no response is requested. The confirmation APDU is sent automatically by the FIS-ground-ASE.

a) the ground CL module generates a FISUplinkAPDU [FISCancelContractsAccept] APDU. It passes the APDU to the ground LI module;

b) the ground LI module then passes the APDU to the air using the D-DATA request. The APDU is passed to the airborne system by way of the upper and lower layers and emerges in the air LI module;

c) the air LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the air CL module;

d) the air CL module recognises the APDU as FISDownlinkAPDU [FISCancelContractsAccept] APDU and passes a FIS-cancel-contracts confirmation to the air HI module. The t-inactivity timer is started; and

e) the air HI module passes the FIS-cancel-contracts confirmation to the FIS-air-user.

5.3.6.1.3 The FIS-air-user specifies the type of FIS contracts it wants to cancel.

5.3.6.2 ***Cancelling a particular Update Contract***

5.3.6.2.1 The *FIS-cancel-update-contract* service allows the FIS-air-user or the FIS-ground-user to cancel in an orderly manner an active FIS update contract. The FIS reports in transit in the communication system are delivered before the FIS update contract is cancelled. This service does not affect the other FIS contracts in place.

a) the FIS-cancel-update-contract request is passed to the local HI module which examines it to see which other module to pass it on to. The request is passed to the local UC module;

b) the local UC module generates a FISDownlinkAPDU or FISUplinkAPDU [FISCancelUpdateContract] APDU. It passes the APDU to the local LI module. Timer t-UC-3 is started to monitor the reception of the reply from the peer ([FISCancelUpdateContractAccept] APDU);

c) the local LI module then passes the APDU to the peer using a D-DATA request. The APDU is passed to the peer system by way of the upper and lower layers and emerges in the remote LI module;

d) the remote LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the remote UC module;

e) the remote UC module recognises the APDU as FISDownlinkAPDU or FISUplinkAPDU [FISCancelUpdateContract] APDU and passes a FIS-cancel-update-contract indication to the remote HI module. If the remote system is air based, the t-inactivity timer is started; and

f) the remote HI module passes the FIS-cancel-update-contract indication to the remote FIS-user.

5.3.6.2.2 The *FIS-cancel-update-contract* service is a confirmed service, initiated by the FIS-air-user or the FIS-ground-user. As the cancellation cannot be rejected by the FIS-ground-user, no response is requested. The confirmation APDU is sent automatically by the FIS ASE.

a) the remote UC module generates a FISUplinkAPDU or FISDownlink [FISCancelUpdateContractAccept] APDU. It passes the APDU to the remote LI module;

b) the remote LI module then passes the APDU to the peer using the D-DATA request. The APDU is passed to the peer system by way of the upper and lower layers and emerges in the local LI module;

c) the local LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the local CL module;

d) the local CL module recognises the APDU as FISDownlinkAPDU or FISUplinkAPDU [FISCancelContractsAccept] APDU and passes a FIS-cancel-contracts confirmation to the local HI module. If the local system is air based, the t-inactivity timer is started; and

e) the local HI module passes the FIS-cancel-contracts confirmation to the local FIS-user.

5.3.7 Aborting a FIS Contract

5.3.7.1 *User-Initiated Abort*

5.3.7.1.1 When a FIS-user detects an error it views as catastrophic, the *FIS-user-abort* service is invoked. All active FIS demand and update contracts processed by the ASE are cancelled. Messages in transit may be lost during this operation. This is an unconfirmed service without parameter.

5.3.7.1.2 The *FIS-user-abort* service can be invoked at any time that the FIS-user is aware that any FIS service is in operation. After this primitive is invoked, no further primitives may be invoked for the current FIS contracts.

a) the FIS-user-abort request is passed to the local HI module which examines it to see which other module to pass it on to. The request is passed to the local AB module;

b) the local AB module requests all running local ASE modules to stop their current activities. It request the local LI module to abort the dialogue;

c) the local LI module abort the dialogue using a D-ABORT request if the dialogue is still open. The abort originator parameter is set to “user”. A D-ABORT indication emerges in the remote LI module;

d) the remote LI module passes the D-ABORT indication to the remote AB module;

e) the remote AB module recognises a user abort situation (the abort originator of the D-ABORT indication is set to "user"). It requests all running remote ASE modules to stop their current activities. It passes a FIS-user-abort indication to the remote HI module; and

f) the remote HI module passes it to the remote FIS-user if still active.

5.3.7.2 *Provider-Initiated Abort*

5.3.7.2.1 When the FIS ASE detects an error from which it cannot recover, the *FIS-provider-abort* service is invoked. This is a FIS provider-initiated service with a single, mandatory parameter telling why the FIS Contracts in progress with a ground FIS system were lost. Messages in transit may be lost during this operation.

a) the local ASE module detecting the error situation requests the local AB module to abort the communication with a reason value;

b) the local AB module requests all running local ASE modules to stop their current activities. It passes a FIS-provider-abort indication to the local HI module. It generates a FISUplinkAPDU or FISDownlinkAPDU [FISAbort]. It passes the APDU to the local LI module;

c) the local HI module passes the FIS-provider-abort indication to the local FIS-user if he is still active;

d) the local LI module passes the APDU to the remote system using a D-ABORT request if the dialogue is still open. The abort originator parameter is set to "provider". The APDU is passed to the remote system by way of the upper and lower layers and emerges in the remote LI module;

e) the remote LI module examines the APDU in order to decide which module to pass it to. The APDU is passed to the remote AB module;

f) the remote AB module recognises a provider abort situation (the abort originator of the D-ABORT indication is set to "provider"). It requests all running remote ASE modules to stop their current activities. It decodes the FISUplinkAPDU [FISAbort] APDU and passes a FIS-provider-abort indication to the remote HI module; and

g) the remote HI module passes it to the remote FIS-user if still active.

5.4 FIS SARPs Section DESCRIPTION

5.4.1 SARPs Section 2.4.2: GENERAL REQUIREMENTS

5.4.1.1 *SARPs Section 2.4.2.1: FIS ASE Version Number*

5.4.1.1.1 SARPs section 2.4.2.1 is included to allow the Context Management (CM) application to exchange version numbers of the FIS Application. It is necessary to allow for future versions of the FIS protocol to be exchanged by the CM Application. It has no effect on the FIS functionality.

5.4.1.2 *SARPs Section 2.4.2.2: Error Processing Requirements*

5.4.1.2.1 In the abstract service definition, each service has a set of parameters and the abstract syntax of those parameters specified. Thus information which is not a valid syntax is not allowed to be input.

5.4.1.2.2 In the protocol definition, there is a requirement that no service is permitted to be called when the ASE is in an inappropriate state. Thus making use of the abstract services is not permitted at these times.

5.4.1.2.3 An implementation does not allow the user to take such invalid actions; however, there is no requirement to prevent an implementation from allowing this. The error processing requirements section thus says that *if* the implementation allows the user to enter invalid information, the system must inform the user that an entry error has occurred. In that case, the error is locally detected and the dialogue does not need to be aborted.

5.4.2 SARPs Section 2.4.3: THE ABSTRACT SERVICE

5.4.2.1 *The Concept of Abstract Service*

5.4.2.1.1 SARPs section 2.4.3 concerns the FIS abstract service. The following paragraphs provide an explanation of the term “Abstract Service”.

5.4.2.1.2 In order to define the FIS ASE (i.e. the module that contains the protocol machine - see SARPs section 2.4.5), it is necessary to describe its reactions to both PDUs arriving from the peer ASE, and the actions from the local FIS-user. The PDUs are well defined in the protocol. The actions of the FIS-user, however, are not. The SARPs do not attempt to dictate the actions of the FIS-user except where absolutely necessary. Despite this, in order to define the ASE it is necessary to have a clear definition of user actions.

5.4.2.1.3 In order to get around this conundrum, an “application abstract service” is defined. The FIS abstract service is a technical description of the interactions between the FIS-user and the ASE. These interactions are precisely defined in SARPs section 2.4.3. Having this definition allows the ASE to be specified precisely in terms of its reactions to the arrival of PDUs and the invocation of the service primitives by the user. This, therefore, is the reason for defining the abstract service.

5.4.2.1.4 The abstract service is defined as being the description of the interface between the ASE and the ASE service-user. These are known as the FIS ASE and the FIS-user. The FIS user is generally not the human user; it is that part of the system that uses the FIS ASE.

5.4.2.2 *The Concept of APIs*

5.4.2.2.1 If one was to buy an FIS application, one would be buying a suite of executable code. From the code itself it is impossible to know whether or not the abstract service has actually been implemented. Therefore the FIS SARPs do not require that the abstract service has to be build has a real interface. It only requires that, when one examines it from an external point of view, it behaves in the same way as if it had been built. This is the explanation of SARPs requirement 2.4.3.1.1.

5.4.2.2.2 Thus the implementors may choose to build a FIS application with an Application Programmatic Interface (API) that corresponds to the FIS application abstract service, or they may choose not to - it is entirely up to them. However, it must be realized that there are a number of good reasons why one might not want to build a system with an API exactly like the abstract service. Examples include:

a) there may be a more efficient way of building the software;

b) the abstract service does not include parameters that are needed locally, but do not effect the state machine; for example, an API might include an indication of which ground system a FIS report has come from;

c) it may not be easy to build the abstract service from the development tools that are used being used; and

d) the abstract service does not have any programming language bindings. An API would require an interface defined in a particular programming language.

5.4.2.2.3 Implementation of the abstract service interface is not mandated by the FIS SARPs. The requirements for FIS set out by ICAO are limited in scope - they are designed only to ensure interoperability between air and ground systems, and to ensure that they meet the stated functionality requirements. The FIS SARPs do not specify the nature of any internal interface within the software, nor do they specify the human interface. Individual implementation projects define their own internal interfaces to suit their own requirements.

5.4.2.2.4 In summary, an application abstract service is defined in the SARPs in order to be able to define the ASE protocol machine. It does not have to be built as an API in any implementation, and there are several good reasons why it is not implemented exactly as defined. A real implementation of the FIS SARPs would normally be expected to define its own APIs.

5.4.2.3 *Functional Model of the FIS Application*

5.4.2.3.1 Figure 5.4-1 below shows an abstract model for the FIS application. Just as with the abstract service, this model shows a design of the FIS application, breaking it down into modules. However, there is no requirement that an implementation actually builds it this way.

5.4.2.3.2 The figure is presented here in order to explain the terms that are used throughout the SARPs. It is not required that the design of an implementation follows this structure. The figure shows three modules:

a) the FIS-user (which could be a FIS air user or a FIS ground user);

b) the control function; and

c) the FIS ASE (application service element — which could be a FIS-air-ASE or a FIS-ground-ASE).

5.4.2.3.3 In addition, it defines the FIS application entity as the control function together with the FIS ASE.

5.4.2.3.4 Abstract interfaces are shown between the different modules:

a) the FIS application entity service interface — which is the same as the abstract service interface defined in SARPs section 2.4.3;

b) the FIS application service element service interface —which is also the same as the

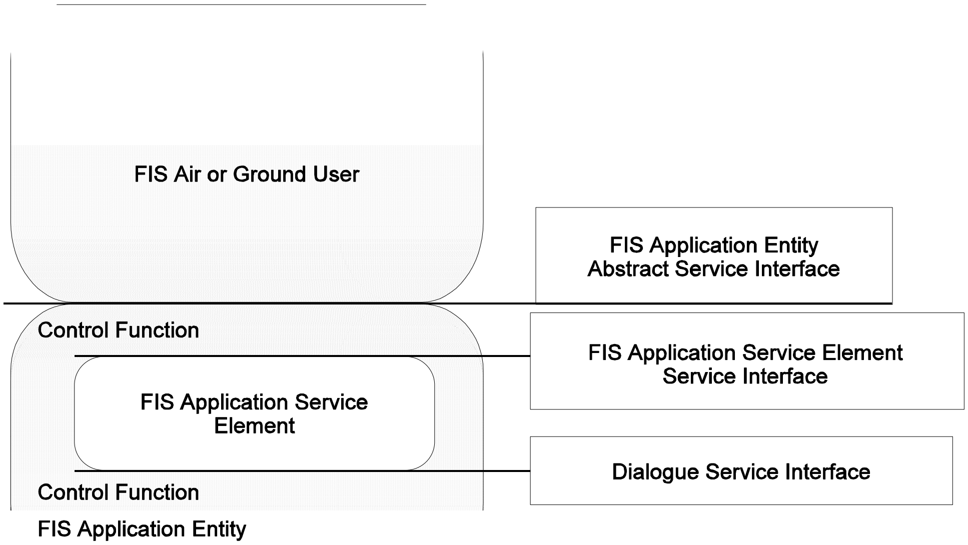


Figure 5.4-1. The FIS Functional Model

abstract service interface defined in SARPs section 2.4.3; and

c) the dialogue service interface — which is defined in the Upper Layer Communications SARPs, and is identical for all air/ground applications.

5.4.2.3.5 Since the FIS application entity service interface is identical with the FIS application service element service interface, the control function module passes primitive calls directly from one to the other without interference.

5.4.2.4 *Conventions for Service Primitives*

5.4.2.4.1 The FIS SARPs define seven FIS services. A FIS service consists of one to four primitives. Each primitive consists of the name of the FIS service (FIS-demand-contract, FIS-update-contract, FIS-cancel-contracts, FIS-cancel-update-contract, FIS-user-abort and FIS-provider-abort) and a suffix that indicates at what point in the service the primitive occurs (request, indication, response, confirmation):

a) the FIS-user that initiates the service calls on the FIS ASE to perform an action — this is called the “request”;

b) after the request is passed to the FIS ASE on the other side of the communication link, it uses the service to pass the information on to its FIS user — this is called the “indication”;

c) the FIS-user that has received the indication may choose to respond to it, in which case it calls upon its FIS ASE to send a reply — this is called the “response”; and

d) finally, the FIS ASE receiving the response provides its FIS-user (which started the sequence of events) with the information — this is called the “confirmation”.

5.4.2.4.2 The terms “request”, “indication”, “response” and “confirmation” come from the field of communication protocols. A given service need not use all four primitives. Some FIS services make use of only one (indication), some two (request and indication, request and confirmation), some three (request, indication and confirmation) and some all four (request, indication, response and confirmation). These primitives are illustrated in Figure 5.4-2.

5.4.2.4.3 A *confirmed FIS service* is one that involves a handshake between the user that requests the service and the user or the ASE that is informed that the service has been requested. Figures 5.4-2 and 5.4-3 illustrate the confirmed FIS services.

5.4.2.4.4 An *unconfirmed FIS service* involves no handshake. Figure 5.4-4 illustrates an unconfirmed service.

5.4.2.4.5 A *provider-initiated service* is generated by the service provider in response to some internal condition. They consist of one indication primitive which is given to both users by the FIS service provider. Figure 5.4-5 illustrates a provider-initiated service.

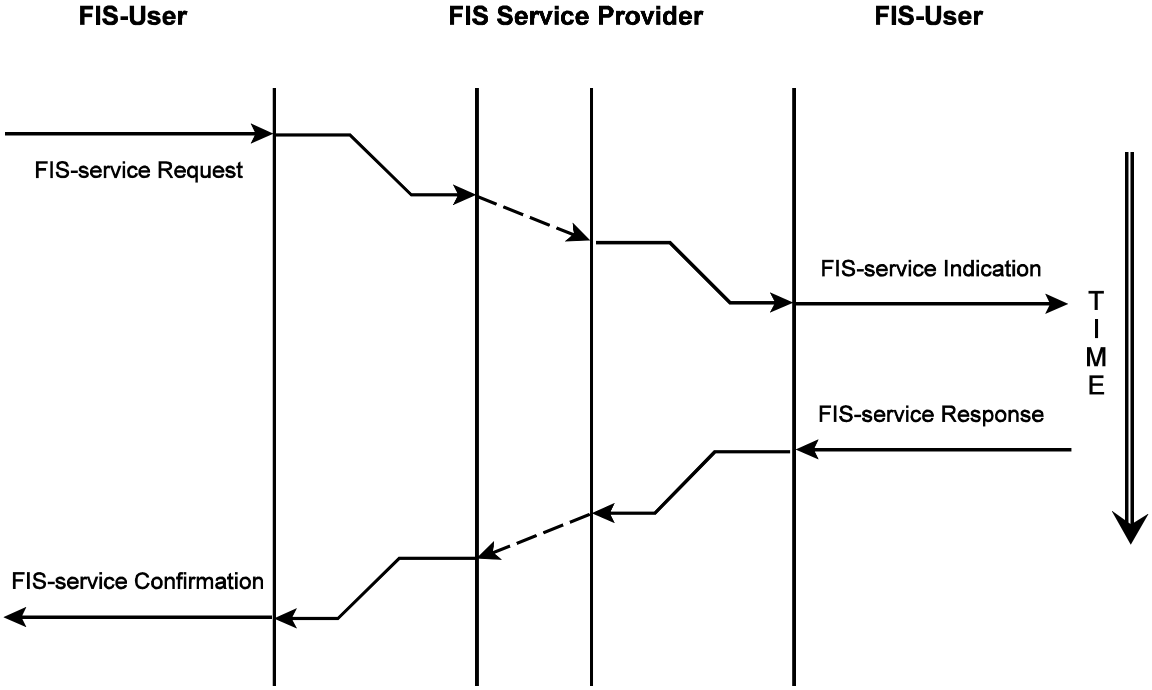


Figure 5.4-2. Generic FIS Service Primitives

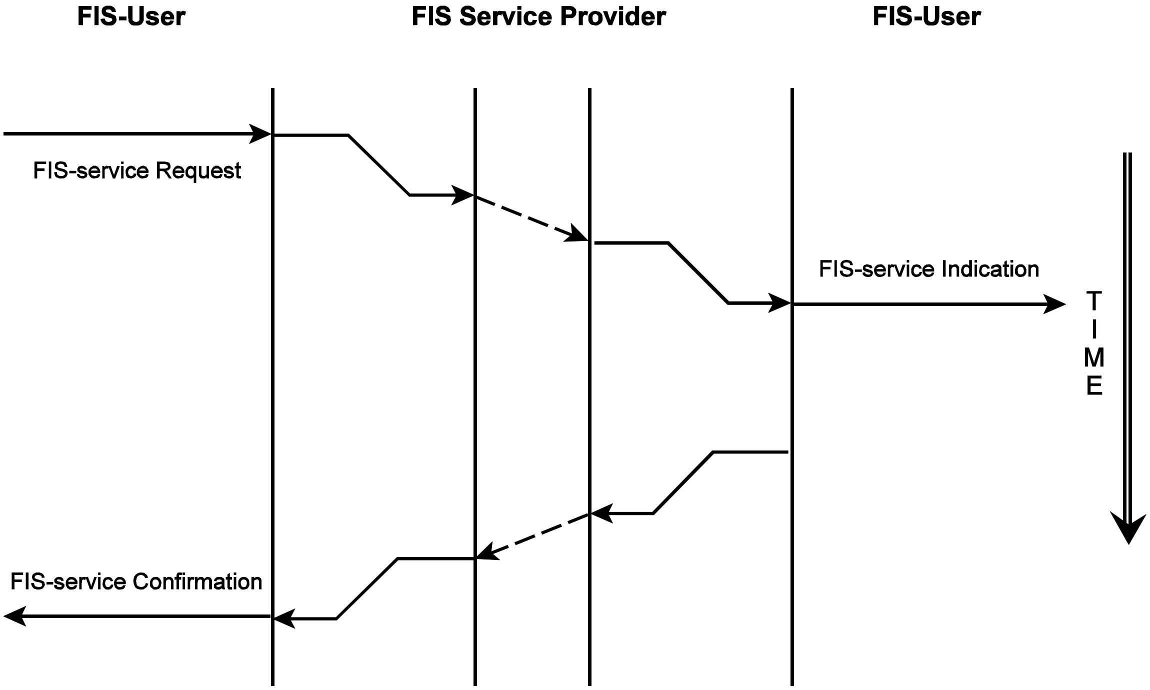


Figure 5.4-3.

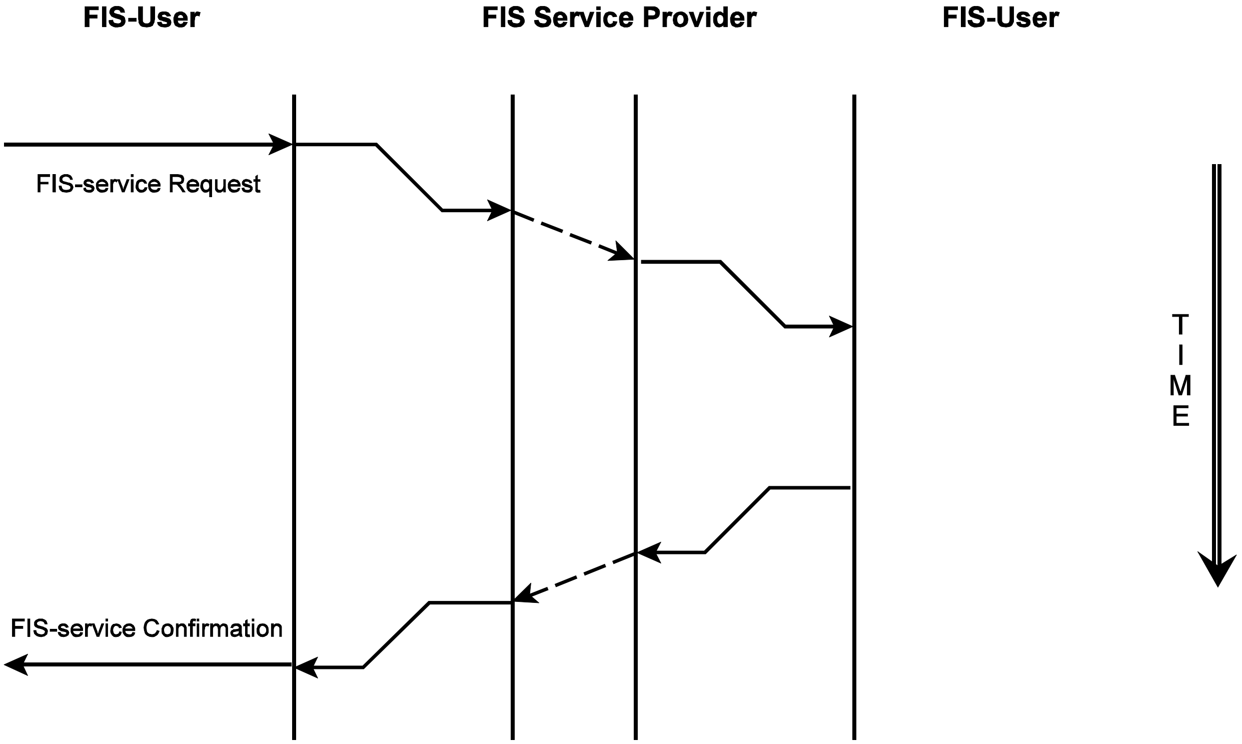


Figure 5.4-3 a and b. Other confirmed FIS services

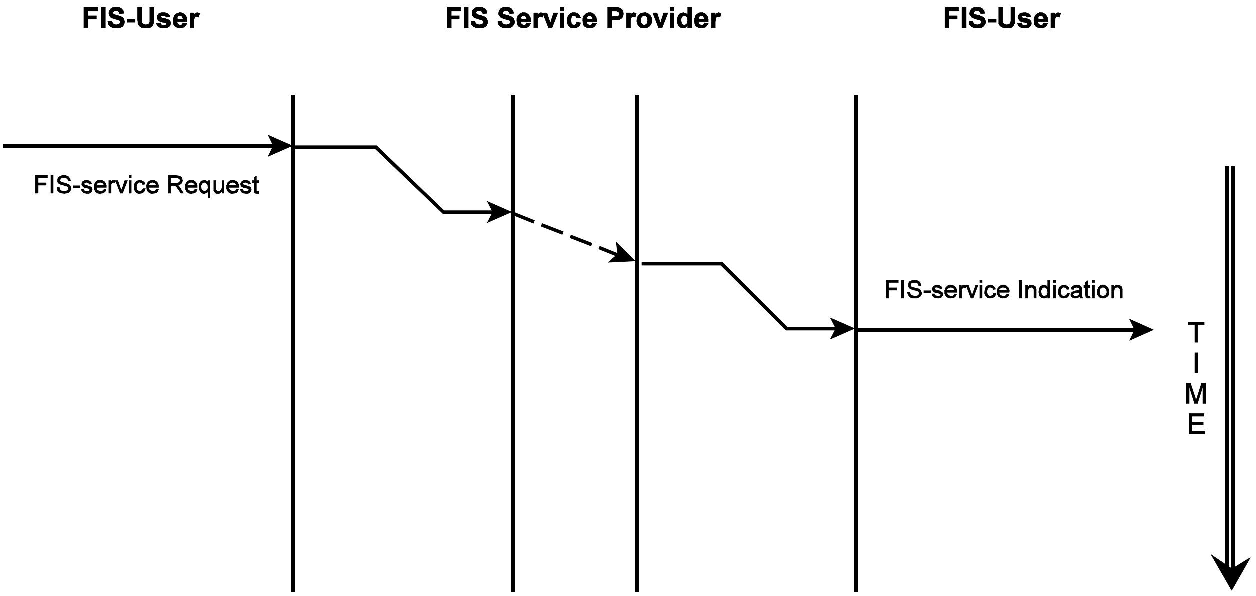


Figure 5.4-4. Unconfirmed Service

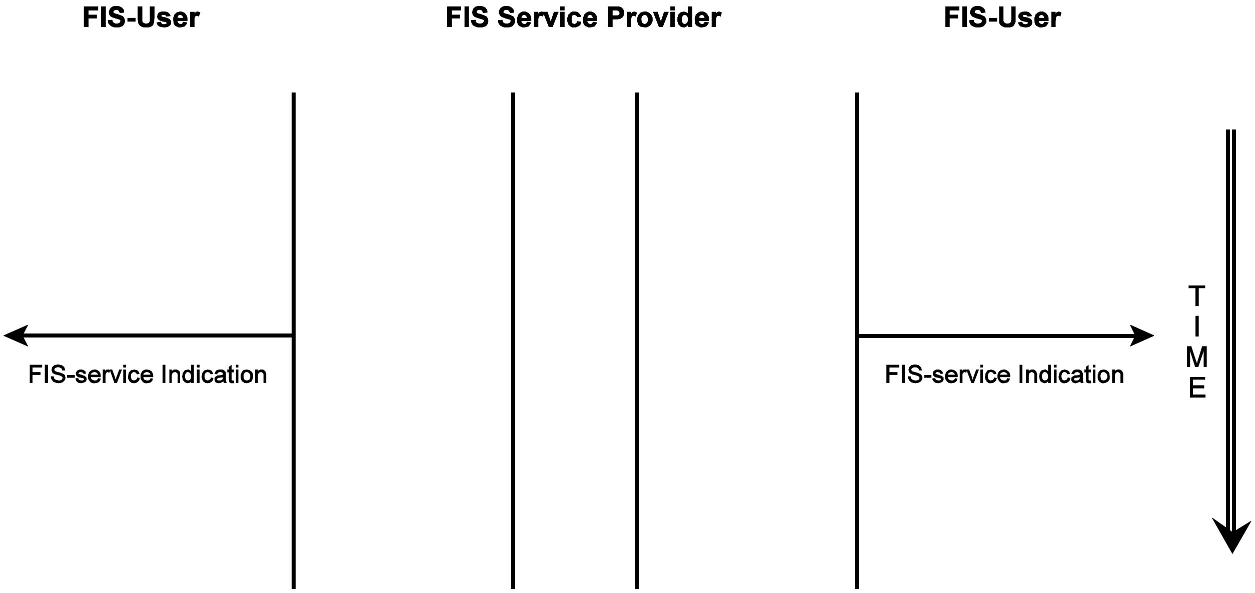


Figure 5.4-5. FIS Service Provider-Initiated Service

5.4.2.5 ***Convention for Service Parameters***

5.4.2.5.1 The services are depicted in the SARPs by primitive and parameter tables as shown in Table 5.4-1. Not all services require each primitive to be used for each point in the service. That is, if indication of a particular primitive is not needed due to reasons like redundant information being relayed, then the parameter column for that primitive is omitted.

Table 5.4-1. Description of the FIS Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
ICAO Facility Designator	M			
FIS Contract Number	M	M(=)	M(=)	M(=)
Class of Communication Service	U			
FIS Contract Details	M	M(=)		
Result			M	M(=)
Reject Reason			C	C(=)
FIS Information			C	C(=)

5.4.2.5.2 For a specific primitive, each parameter is described by a value that dictates the terms under which that parameter is used. If the use of any parameter does not follow the rules as set forth by the primitive and parameter tables, there is an error in the implementation. The abbreviations used in the primitive and parameter sections are described in the following sections.

a) a "C" is used to indicate that the parameter is conditional upon some state. A "C" differs from a "U" (User Option) due to the fact that if the stated condition exists, the parameter must be supplied, while a "U" means that the parameter's use is wholly up to the user;

b) a "C(=)" is used to show that a parameter is conditional upon the value of the parameter to the left of it in the table being present, and if it is present, the "C(=)" parameter will retain the parameter to the left in the tables's value;

c) a "M" is used to indicate that the use of that parameter is mandatory, and no option not to use it exists;

d) a "M(=)" is used to indicate that the parameter will always take the value to the left of it in the table; its presence is not conditional;

e) a "U" is used to indicate that the use of the parameter is a user option. Therefore the presence of the parameter is optional, and will be used based upon a user requirement. Some of the requirements for "U" parameters are specified in SARPs section 2.4.7. The use of "U" parameters is not wholly specified in the SARPs. Further requirements determining the use of "U" parameters may be determined by operational or procedural requirements outside the scope of the CPDLC SARPs; and

f) a blank in the table is used to indicate that the specific parameter will not be used.

5.4.2.5.3 When no parameter is specified in the table (e.g. the FIS-user-abort service), this means that the service has the primitive listed in the table but not parameter are needed.

5.4.2.5.4 If a parameter column for a primitive is present, but all of the parameters are left blank, that means that the primitive is defined for the service but does not contain any parameter.

5.4.2.5.5 Note also that not all parameters need to present for all indications of a particular primitive.

5.4.2.5.6 Throughout the service descriptions every parameter is described. In particular, there is a note that explains the purpose of the parameter, and a requirement that states what values it may contain.

5.4.2.5.7 In many cases, the primitive parameters have the same contents than APDU fields in the ASN.1 description. The FIS ASE is tasked to copy the parameter value within the APDU field without modification. In order to avoid confusion by defining twice identical data structures, the type of those primitive parameters is specified by simply referring to the corresponding ASN.1 type in the APDU. The ASN.1 is used in the service definition as a syntax notation only and does not implicitly imply any local encoding of these parameters. The local implementation of these parameters remains a local implementation issue.

5.4.2.5.8 For the other parameters, the syntax is described by enumerating the authorised abstract values.

5.4.2.6 *FIS Services*

5.4.2.6.1 This section lists the FIS services, and explains why each is a confirmed or an unconfirmed service.

5.4.2.6.2 The FIS-demand-contract and FIS-update-contract services are both confirmed since an explicit acceptation or rejection of the contract is expected from the peer.

5.4.2.6.3 The FIS-report service is unconfirmed, since there is no requirement for the ground system to know when the airborne system has received the report.

5.4.2.6.4 The FIS-cancel-update-contract service and the FIS-cancel-contracts services are confirmed since these services may constitute the last exchange between the initiator and the responder. The initiator needs to know when the exchange is definitively closed (in order for instance to free all the resources allocated for the communication).

5.4.2.6.5 The FIS-user-abort service is an unconfirmed service since there is no requirement for the initiating system to know that the abort has arrived at the peer.

5.4.3 **SARPs Section 2.4.4: FORMAL DEFINITION OF MESSAGES**

5.4.3.1 *Introduction*

5.4.3.1.1 The FIS abstract syntax is written in a notation that is called ASN.1. It is strongly recommended that the reader is familiar with ASN.1 in order to understand the details of this SARPs section.

5.4.3.2 *SARPs Section 2.4.4.1: Encoding/Decoding Rules*

5.4.3.2.1 SARPs section 2.4.4.1 defines which APDUs the FIS systems must be able to encode and decode.

5.4.3.2.2 An APDU (Application Protocol Data Unit) is a sequence of bits that are passed from one peer application to another. The sequence of bits is a concrete representation of a message that is passed between applications. An APDU for this message will be a sequence of bits representing the information to be carried.

5.4.3.2.3 A FIS system is not required to be able to encode and decode all messages specified in the ASN.1 description. For instance, an air FIS system is not required to be able to encode an FISUplinkAPDU APDU nor to decode an FISDownlinkAPDU APDU.

5.4.3.3 *SARPs Section 2.4.4.2: FIS ASN.1 Abstract Syntax*

5.4.3.3.1 SARPs section 2.4.4.2 defines the abstract syntax of the protocol. That is, it defines the structure of the PDUs that are to be sent between aircraft and the ground systems

5.4.3.3.2 Data types exchanged by FIS ASEs are described in the FIS SARPs by using a machine-independent and language-independent syntax. There is no constraint put on the implementors concerning the machine nor the development language to be selected for implementing the protocol.

5.4.3.3.3 The ASN.1 module *MessageSetVersion1* contains the data types of the protocol data units handled by the FIS ASEs. Unlike common OSI ASEs (e.g. ACSE), no object identifier has been attached to the FIS ASN.1 specification. Indeed, the ULCS architecture releases the applications from negotiating during the dialogue establishment of the applicable abstract syntax. Object identifiers related to FIS applications (application context name and version number) are defined in the ULCS SARPs.

5.4.3.4 *ASN.1 Organization*

5.4.3.4.1 The ASN.1 itself is organized into a number of different sections:

a) the FIS messages — top level;

b) the FIS messages — second level;

c) the ATIS messages; and

d) the ATIS fields.

5.4.3.4.2 An index of the types defined in the ASN.1 is given in section 5.6.

5.4.3.5 *ASN.1 Tags*

5.4.3.5.1 Tags are used in ASN.1 to allow distinguishing data types when confusion is possible. For instance, when a data type contains two optional elements of the same type, if only one is encoded there is no means for the decoder to know which element the decoded value is attached to.

5.4.3.5.2 Even if tag values are not used by the Packed Encoding Rules, the ASN.1 grammar mandates the use of tags in some cases. When specifying the FIS data types, the following rules have been used:

a) tags are always used within CHOICE data type, starting at 0 and then incremented by 1 for each entry; and

b) tags are not used at all in SEQUENCE data types when no confusion is possible. When an optional element is defined, all elements in the sequence are tagged.

5.4.3.6 ***Extensibility Markers***

5.4.3.6.1 In order to allow the upgrade of the ASN.1 specification when new FIS services are made available, the extensibility ASN.1 feature (ellipse) has been used in each data type identifying the DFIS service.

5.4.3.6.2 For instance, extensions have been allowed in the following data types:

a) "DownlinkAPDU" to allow the definition of new FIS downlink APDUs;

b) "UplinkAPDU" to allow the definition of new FIS uplink APDUs;

c) "FISAbort" to allow the definition of abort data for new FIS services; and

d) "FISCancelAcceptData" to allow the definition of cancel accept data for new FIS services, etc.

5.4.3.7 ***Handling Numerical Values***

5.4.3.7.1 Some operational data are real. REAL data types exist in ASN.1 but the associated encoding procedures are not optimized in term of data size (the Packet Encoding Rules do not specify specific rules for real, but import the Basic Encoding Rules ones which require the encoding of a mantissa, a base and a exponent taking several octets). Real values are therefore specified via INTEGER data type.

5.4.3.7.2 ASN.1 constraints are defined for INTEGER data types to take advantage of both range and resolution. Lower bound is equal to the operational minimal value devised by the resolution and the upper bound is equal to the operational maximal value devised by the resolution.

5.4.3.7.3 For instance:

```
PressureMeasure ::= CHOICE
{
hPa[0] INTEGER (7500..12500)
units = hPa, range (750.0..1250.0), resolution 0.1
inches[1] INTEGER (2200..3200)
units = inches of Mercury, range (22.00..32.00), resolution = 0.01
}
```

5.4.3.7.4 This definition of real fields allows for a very simple algorithm for computing the operational value from the encoded value and vice-versa. The message sender divides the operational value (e.g. PressureMeasure = 25.54) by the resolution (0.01) to find the value to send (2554) whereas the receiver multiplies the received value by the resolution to find the operational value.

5.4.3.8 *ASN.1 Entry Points*

5.4.3.8.1 The top level (which will typically be used as entry point in any ASN.1 compiler) consists of two main structures: the FISDownlinkAPDU is a choice of time-stamped PDUs generated by the aircraft, and FISUplinkAPDU is a choice of time-stamped PDUs generated by the ground system.

5.4.3.9 *Time Representation*

5.4.3.9.1 Data types have been specified for containing time indication (Date, DateTimeGroup, Year, Month, Hours, Minutes, and Seconds). This way of representing time is preferred over the pre-defined ASN.1 representations (GeneralizedTime and UTCTime) for optimization of the PER encoding.

5.4.3.10 *Guidances for building FIS Reports*

5.4.3.10.1 The ASN.1 notation allows for a strict and unambiguous specification of data structures. Data sent on the line are checked before transmission as being compliant with the structure defined in the ASN.1. However, a great level of flexibility is left to the application-users when filling in the data to be sent and it is of their responsibility to check that the data structures they build are operationally valid.

5.4.3.10.2 Indeed, a lot of fields are defined as optional but the conditions of presence and the valid combinations of fields are not described at all in the ASN.1 specification. Another example is the CHOICE type where the selected component could be chosen according to the value of an other field. The reason for not including these conditions in the ASN.1 is that this would cause an unnecessary complexity of the ASN.1.

5.4.3.10.3 For instance, in an ATIS report, the CAVOK indication shall replace the specification of Clouds, Present Weather and Visibility. It is meaningless to have in the same report both the CAVOK indication and the description of the visibility. However, the current ASN.1 allows technically to build an ATIS report including CAVOK, Present Weather and Visibility.

5.4.3.10.4 The FIS application-user is therefore requested to follow the construction rules of FIS messages expressed in the relevant ICAO documents, as for ATIS the annexes 3, 4, 5, 11, 14 and 15. The FIS-ASEs do not check the semantics of the messages exchanged.

5.4.4 **SARPs Section 2.4.5: PROTOCOL DEFINITION**

5.4.4.1 *SARPs Section 2.5.5.1: Sequence Rules*

5.4.4.1.1 Time sequence diagrams or message sequence diagrams are used to denote the relationship between the primitives that form a FIS service and the order in which they occur, e.g. the indication/confirmation primitives occurs some time after the request/response primitives.

5.4.4.1.2 Inherent to the service model is the notion of queuing, as illustrated in Figure 5.4-6. The FIS-service indications and confirmations are delivered to the FIS-users in the order that the corresponding FIS-service requests and responses were issued. One exception to the notion of queuing is the abortive services (FIS-user-abort and FIS-provider-abort services) which may overtake other primitives and empty the primitives in the queue.

5.4.4.1.3 Each figure in SARPs section 2.4.5.1 has the same structure as illustrated in Figure 5.4-7. There are four vertical lines that separate the five major components in the FIS system. From left to right, they are:

a) the FIS-ground-user — that part of the ground system that processes the FIS information;

b) the FIS-ground-ASE — that part of the ground system that implements the FIS protocol;

c) the dialogue service provider — that part of the ground system, the air system and the networks that, together, provide the dialogue service, as defined in the upper layer architecture — on the figures this is the thin strip down the middle;

d) the FIS-air-ASE — that part of the air system that implement the FIS protocol; and

e) the FIS-air-user — that part of that uses the FIS service to provide information to human users.

5.4.4.1.4 The middle three sections of the diagrams (FIS-ground-ASE, dialogue service provider and FIS-air-ASE) together form the FIS service provider, and are labelled as such on the diagrams.

5.4.4.1.5 The outer two vertical lines represent the FIS abstract service. Any lines crossing them represent the invocation of one of the FIS service primitives. The FIS service primitives are labelled in the FIS user part of the figure.

5.4.4.1.6 The inner two vertical lines represent the Dialogue service. Any lines crossing them represent the invocation of one of the Dialogue service primitives. The Dialogue service primitives are labelled in the FIS ASE part of the figure.

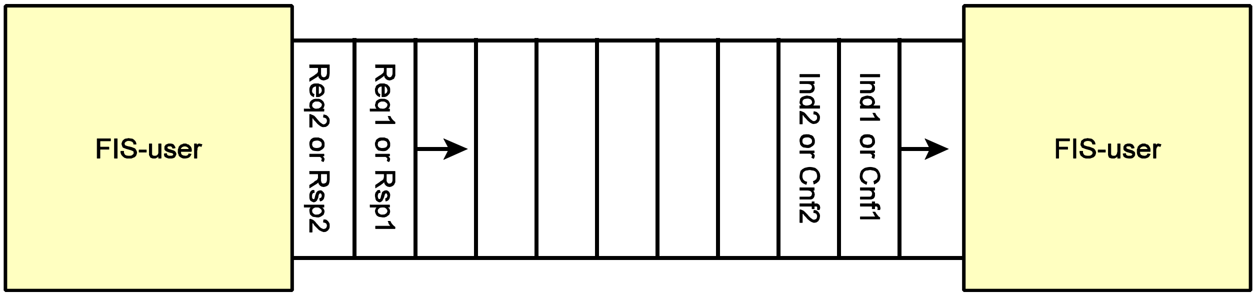


Figure 5.4-6. Communication Queue

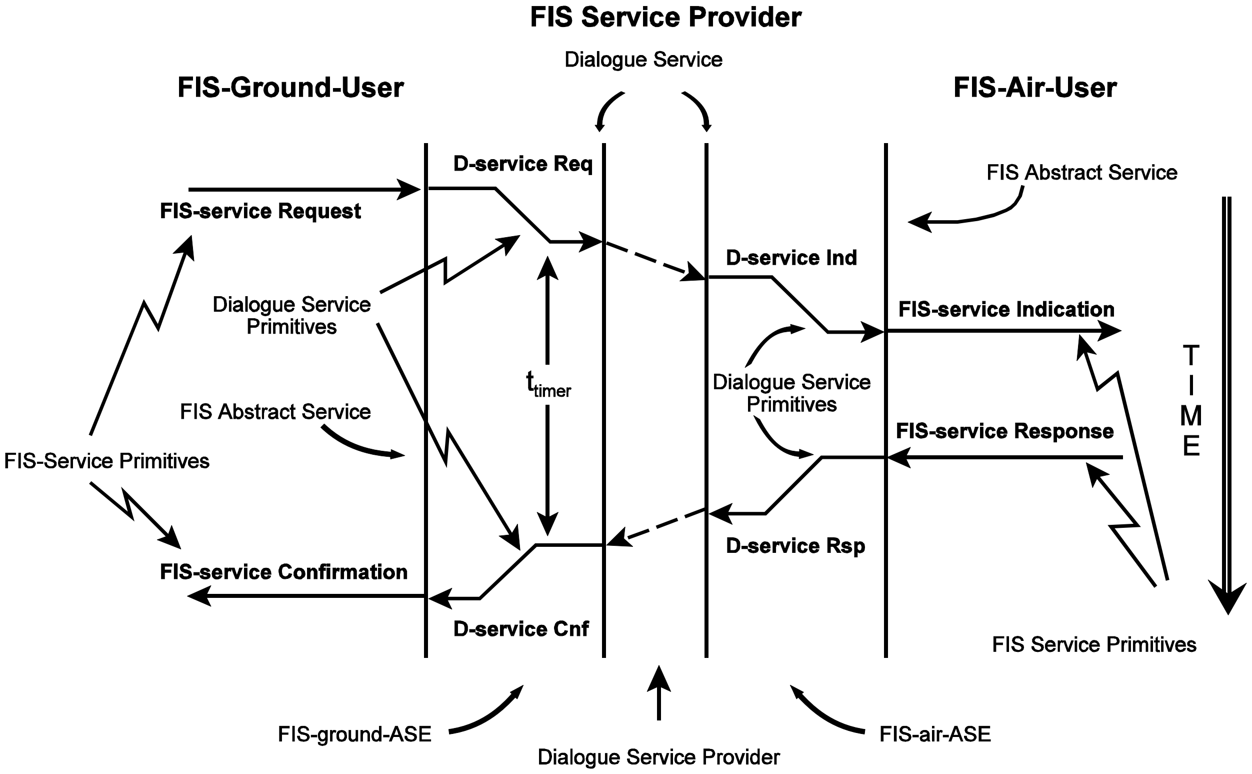


Figure 5.4-7. Message Sequence Diagram

5.4.4.1.7 The diagrams represent a sequence of events. Time is always considered to run down the figure from the top (representing the earliest time) to the bottom (representing the latest time).

5.4.4.1.8 If the ASEs set timers, there are marked on the figures by vertical lines with arrows at both ends. This is depicted in Figure 5.4-7 by the "t_{timer}" label.

5.4.4.1.9 Note that the last figures representing an abort situation can be overlaid on top of any of the other figures to represent an abort in action.

5.4.4.2 *SARPs Section 2.4.5.2: FIS Service Provider Timers*

5.4.4.2.1 SARPs section 2.4.5.2 lists the service provider timers that are defined in the FIS protocol, and suggests values for them.

5.4.4.2.2 The purpose of the service provider timers in the FIS service provider is not operational. For operational reasons, there may be a requirement to have other timers that are shorter than the ones described here. The purpose of these timers is only to ensure that the ASE protects itself when communicating with a system that has failed, for some reason, to respond.

5.4.4.2.3 For example, suppose an air system sends a FIS-demand contract request to a ground FIS server. Suppose further that the ground system has not implemented the FIS protocol correctly, and its locks up without sending the FIS demand response to the aircraft. The airborne system will be in a state that is waiting for a result. The specification prevents the air system from sending down another message until the first has been dealt with. This combination of events would make the air system lock up unless a timer is used. When in this state, the t-DC-1 timer will eventually reach its maximum value. The air system can then abort the connection. Thus the air system protects itself from getting indefinitely locked up because of another system's failure.

5.4.4.2.4 Noted that the values set in the FIS service provider timers have been calculated thus: it has been assumed that the slowest possible network is in operation and that all systems in the path have their maximum delay. Does a reply to a request take longer than this, then it is assumed that something must be failing somewhere.

5.4.4.2.5 The assignment of values for timers must be optimised based on operational testing of the application. In such testing, incompatible timer values and optimum combinations can be identified. Implementations of FIS protocol are required to support configurable values for all timers and protocol parameters, rather than having fixed values. This allows modification as operational experience is gained.

5.4.4.3 *SARPs Section 2.4.5.3: FIS-ASE Protocol Description*

5.4.4.3.1 The FIS service provider is described in the SARPs as finite state machines or protocol machines (PM). The protocol machine for a particular service starts in an initial state (IDLE). Events, which are service primitives received from the FIS-users above or the Dialogue Service provider below, as they occur, trigger activity on the part of the PM. As part of this activity, actions may be required (service primitives issued to the FIS-users and/or the underlying Dialogue Service provider).

5.4.4.3.2 The protocol description explains the rules by which the ASEs work. There is a detailed specification of actions taken by the ASEs when triggered by certain events:

a) the arrival of a PDU through the dialogue service;

b) the invocation of one of the service primitives by the user;

c) the expiration of one of the internal timers; and

d) the occurrence of an unrecoverable error.

5.4.4.3.3 The protocol is described under two forms: the textual and the tabular descriptions. The textual description takes precedence over the tabular description.

5.4.4.3.4 In order to resolve a somewhat complex specification, both the ground and the air ASE have been broken up into a number of internal modules. The behavior of each is specified separately. In addition to the four triggers specified above, individual modules may also be triggered into action by the other modules within the same ASE.

5.4.4.3.5 Both air and ground ASEs have modules that mirror each other. However, apart from the AB and LI modules, the actions of the air and ground mirrored modules are different but complementary. The modules are:

a) HI module — the main job of the HI module is to select which module will handle the primitives that are invoked by the user;

b) LI module — the main jobs of the LI module are a) to select which module handles the PDU passed to it from the dialogue service, and b) to manage the dialogue, selecting which dialogue service is to be used at any given time;

c) DC module — the job of the DC module is to manage demand contracts;

d) UC modules — the job of the UC module is to manage update contracts;

e) CL module — the job of the CL module is t manage the cancellation of contracts; and

f) AB module — the job of the AB module is to handle abort situations.

5.4.4.3.6 There is a requirement that the ASE does not accept the invocation of primitives when no actions are described for that primitive in that state (2.4.5.3.2). Some explanation of this statement is needed. Each module has several different states. When a module is in a particular state, only some primitives are permitted to be invoked. For example, if a FIS-update-contract request is invoked, it is not permitted to invoke a second FIS-update-contract request until a reply has been received for the first one. There is no statement in the description of the protocol that explains what the ground ASE does if it receives a second FIS-update-contract request before the reply to the first one has been received. The SARPs therefore require (by statement 2.4.5.3.2), that the FIS-air-user must not invoke a FIS-update-contract request during this period.

5.4.4.3.7 Thus only actions which are permitted are described. If an action is not described, then it is not permitted.

5.4.4.3.8 Actions are enumerated in the order they have to be carried out by the protocol machine.

5.4.4.4 *SARPs Section 2.4.5.5: State Tables*

5.4.4.4.1 State tables are provided in SARPs section 2.4.5.5. These are an exact reflection of the FIS protocol description, in a condensed form.

5.4.4.4.2 The state tables are only presented for guidance, since the textual protocol description always takes precedence.

5.4.5 SARPs Section 2.4.6: COMMUNICATION REQUIREMENTS

5.4.5.1 *SARPs Section 2.4.6.1: Encoding Rules*

5.4.5.1.1 SARPs section 2.4.6.1 states that PER (Packed Encoding Rules) must be used to encode the PDUs. PER is an ISO standard and is particularly efficient at encoding data. Implementors may use ASN.1 compilers to generate code that encode and decode PER automatically.

5.4.5.2 *SARPs Section 2.4.6.2: Dialogue Service Requirements*

5.4.5.2.1 The dialogue service requires a number of parameters to operate. SARPs section 2.4.6.2 defines those parameters that are not defined elsewhere.

5.4.5.2.2 In addition to the *ClassOfCommunication* parameter provided by the contract initiator, other Quality of Service (QOS) parameters are attached by the ASE to the dialogue supporting the communication. Values for the Application Priority and the Residual Error Rate are constant for the FIS application and therefore are not requested to the users.

5.4.5.2.3 The Application Priority for ATIS is set by the ASE to the abstract value " Aeronautical Information Service messages". This requirement follows from the specification in [3] for the ATIS service.

5.4.5.2.4 The Residual Error Rate is set by the ASE to the abstract value "low". This requirement leads to the use of a checksum at the Transport layer, thereby increasing the integrity of FIS exchange.

5.4.5.2.5 No security parameter is requested for the established dialogue.

5.4.5.2.6 The dialogue integrity is provided by the underlying communication service.

5.4.5.2.7 The underlying layers provide a checksum and realise the requirement for message integrity. It is not necessary, therefore, to provide application level integrity by means such as redundancy checks and confirmation of services.

5.4.6 SARPs Section 2.4.7: FIS USER REQUIREMENTS

5.4.6.1 The requirements set out in SARPs section 2.4.7 are in line with those specified in [3]. These requirements are necessary for proper interoperability of ATN applications, and go into more technical detail than those specified in [3].

5.4.6.2 SARPs section 2.4.7 also recommends values for some operational response time which are in line with the values specified for the technical timers:

5.4.6.2.1 Upon reception of a FIS service indication requiring a response, the FIS-user must invoke the response within 1 second.

5.4.6.2.2 Upon receipt of the request to send a FIS report, if the FIS report is not available, the FIS-ground-user must send a positive acknowledgement within 1 second and then send the requested FIS report within 10 seconds.

5.4.7 SARPs Section 2.4.8: SUBSETTING RULES

5.4.7.1 *General*

5.4.7.1.1 There is some functionality within the FIS SARPs that implementors may choose not to incorporate. For example, if a particular implementation of an air system is designed always to use the demand contract, then there is no requirement for it to have the code for update contract implemented.

5.4.7.1.2 There are some combinations of functionality that allow interoperability, and some that do not. SARPs section 2.4.8 defines the combinations of functionality that allow interoperability.

5.4.7.1.3 The combinations of functionality are defined in a set of tables.

a) version number — only one version is defined. This is a placeholder for when future versions are defined;

b) protocol options — this defines a number of options for parts of the protocol that may be implemented. The options may be implemented together. Each has a name associated with it — the predicate;

c) FIS-ground-ASE configurations — this defines two combinations of protocol options, each of which yields a coherent protocol;

d) FIS-air-ASE configurations — this defines two combinations of protocol options, each of which yields a coherent protocol;

e) supported FIS service primitives — this defines the conditions under which the service primitives are applicable; and

f) supported FIS APDUs — this defines the conditions under which the PDUs are applicable.

5.4.7.1.4 The subsetting rules define those subsets that are technically possible. The SARPs do not address the operational acceptability of these subsets.

5.4.7.1.5 Functionality supported by an air-based FIS system is different from the functionality supported by a ground-based FIS system. The air-based FIS system acts as the originator of the FIS contracts whereas the ground-based FIS system is the provider of the FIS information.

5.4.7.1.6 Any aircraft can independently select a subset and be inter-operable with any ground system independently selecting a subset.

5.4.7.1.7 The four configurations defined in the FIS SARPs are illustrated in Figure 5.4-8.

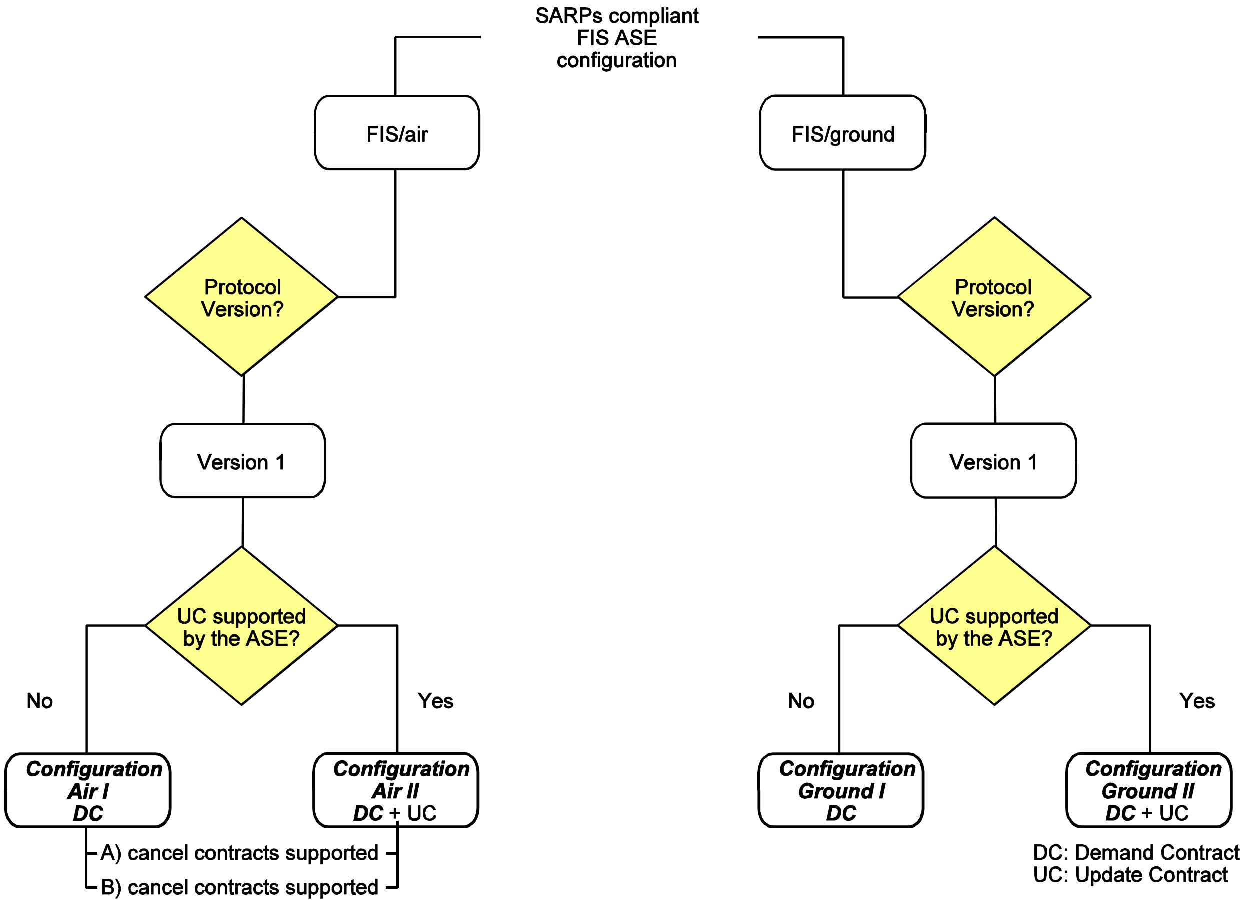


Figure 5.4-8. FIS Subsetting Rules

5.4.7.2 *Mandatory Functionality*

5.4.7.2.1 The FIS-demand-contract service, the FIS-report service and the FIS-abort service are always supported by both air and ground systems. These functions constitute the core functionality of the FIS protocol.

5.4.7.3 *Optional Functionality*

5.4.7.3.1 The FIS update contract service is optionally supported by the FIS systems.

5.4.7.3.2 The FIS update contract is a facility provided to the air-users upon the choice of the implementors. When the FIS-update-contract service is not available to the pilots, they are not automatically warned of the update of the information they have received previously. They have therefore to request the update of the FIS information by invoking the FIS-demand-contract service several times. Another consequence is that the underlying dialogue is not maintained between the aircraft and the FIS ground system and each new FIS demand request causes the establishment of a new dialogue.

5.4.7.3.3 The ground FIS systems provide the FIS update contract service upon the choice of the FIS service provider. However, the ground system must support a stub of the FIS update contract function to enable rejection of the request for an update contract.

5.4.7.3.4 In the case the update contract service is not provided by the ground FIS system, upon receipt of a FIS-update-contract request from an aircraft, the ground ASE forwards the request to the FIS-ground-user. The FIS-update-contract response primitive allows the FIS-ground-user to indicate that it cannot provide updates of the FIS information and optionally to uplink a single FIS report.

5.4.7.3.5 The FIS-cancel-update service is supported only when the update contract is supported.

5.4.7.3.6 The FIS-cancel-contracts service is optionally offered to the air-users. If this service is not available, the air users must cancel individually each FIS contract in sequence to get the same result. A FIS ground system – supporting the FIS update contract or not – is always able to receive a request of cancellation of all contracts.

5.5 Dimensions

5.5.1 PDU Size

5.5.1.1 Theoretically, the FIS ground system could be requested to provide a FIS report with all possible types of information, including all fields of arrival and departure ATIS related to a given airport. The largest PDU that the ground system will be required to produce will thus depend on the maximum number of runways available and the level of use of the free text fields; Where the 36 arrival runways and 36 departure runways authorized by the ASN.1 are used and all free text fields are used at their maximum length, then the largest PDU that the ground system could encode in PER is approximately 28 Kilobytes.

Note.— The computation is performed with the worst technical conditions which have no operational meaning. The ATIS report contains an arrival ATIS for 36 runways and a departure ATIS for 36 runways. For each runway is specified the surface conditions (257b), the braking actions (105b), the arresting systems (1b) and the RVR (6b). Are added the surface wind (5b), the visibility (1b), the cloud (11b), air and dew point temperatures (2b),the altimeter setting (94b), the present weather (2b), the significant met phenomena (387b), the holding delay (200b), the specific ATIS instructions (65b), the other operational information (251b), the transition level (1b) and, for arrival only, the trend type landing forecast (257b).

5.5.1.2 The size of other air and ground initiated PDUs are small by comparison, and all ground systems must be built with the capability to receive all requests from the aircraft.

5.5.1.3 Typical sizes of typical FIS PDUs are indicated in Table 5.5-1.

5.5.1.4 The message size discussed above is well beyond the message size taken into account to determine the message integrity figures given in table 3.1 of [4].

5.5.1.5 Does either airborne or ground system receive a PDU that is too large for it to manage, it will be unable to decode it. The system must abort the connection under these circumstances (with abort reason set to “unrecoverable error”).

Table 5.5-1. FIS PDU Sizes

FIS PDU	Min Size in bytes	Typical Size in bytes	Max Size in bytes
FIS Downlink APDU [FISRequest]	11	11	?
FIS Downlink APDU [FISCancelUpdateContract]	6	6	6
FIS Downlink APDU [FISCancelUpdateAccept]	6	6	6
FIS Downlink APDU [FISCancelContracts]	6	6	6
FIS Downlink APDU [FISAbort]	6	6	6
FIS Uplink APDU [FISAccept]	6 (positive acknowledgement)	30-50 (with a report)	6 (with a report)
FIS Uplink APDU [FISReject]	6 (update function not supported with no report)	7 (other reasons)	28K (update function not supported with a report)
FIS Uplink APDU [FISReport]	?	30-50	28K
FIS Uplink APDU [CancelUpdateContract]	6	6	6
FIS Uplink APDU [CancelUpdateAccept]	6	6	6
Fis Uplink APDU [FISAbort]	6	6	6

5.5.2 FIS Message Frequency

5.5.2.1 The maximum message transmission rate can be reached for an FIS Update Contract with a frequent update of the FIS information. For ATIS, under exceptional conditions, several FIS reports could be sent at a rate of 5 minutes for a short period of time, including the time needed for the operator to input free text data. In normal conditions, an average period between ATIS reports is one hour. This maximum transmission rate does not put strong constraints on the air communication system.

5.5.2.2 However, the FIS data is fairly long-lived but the air-user population changes quickly. A same FIS report will therefore to be sent to all aircraft contacting the ground system.

5.5.3 Number of Concurrent FIS Connections

5.5.3.1 There is nothing in the FIS SARPs that restricts or dictates the number of connections that must be supported, either in the air or on the ground FIS system.

5.5.3.2 *Airborne systems*

5.5.3.2.1 However, dimensioning and cost consideration must be given when allowing an airborne system to run multiple FIS contracts with several ground systems. There will be as many transport connections set up in parallel, as there are contacted ground systems. When parallel contracts are set up with the same ground system, only one transport connection is used and all contracts are multiplexed over this dialogue.

5.5.3.2.2 The number of connections will depend essentially of the deployment of the FIS systems on the ground. A single connection established between the aircraft and a ground FIS server will support all the contracts requested by the pilot for all types of FIS contracts. If the Flight Information Service provider has preferred to install several ground FIS system specialized each in one FIS service, a distinct connection with these grounds systems is needed when contracts are set up in parallel.

5.5.3.3 *Ground FIS systems*

5.5.3.3.1 For the ground FIS system, the number of connections will depend on the number of aircraft likely to establish at the same time an FIS contract for a given airport. In addition, it will depend, for ATIS server, on the coverage of the server in terms of airports. More airports are covered by the server, more aircraft will contact this server.

5.6 ASN.1 INDEX

5.6.1 ASN.1 Section Index

5.6.1.1 Within the main body of the ASN.1, there are four sections divided up by comments crossing the page. These have been given a section number S1 - S4, and are listed in Table 5.6-1 below. These section numbers are not in the SARPs themselves, but have been included here to make the reading of the table in the following section easier. They can be used to determine the location of the ASN.1 type definition within the SARPs.

Table 5.6-1. ASN.1 Sections

Section Reference	ASN.1 Section
S1	FIS messages (top level)
S2	FIS messages (second level)
S3	ATIS messages
S4	ATIS fields

5.6.2 ASN.1 Graphical Representation

5.6.2.1 This section aims at helping the reader to visualize the ASN.1 structures exchanged between the FIS-air-ASE and the FIS-ground-ASE. Data items are represented as boxes containing the ASN.1 name and type of the data. Doted boxes represent optional data, boxes in relief represent terminal data.

5.6.2.2 Figure 5.6-1 represents the top-level APDU generated by the FIS-air-ASE.

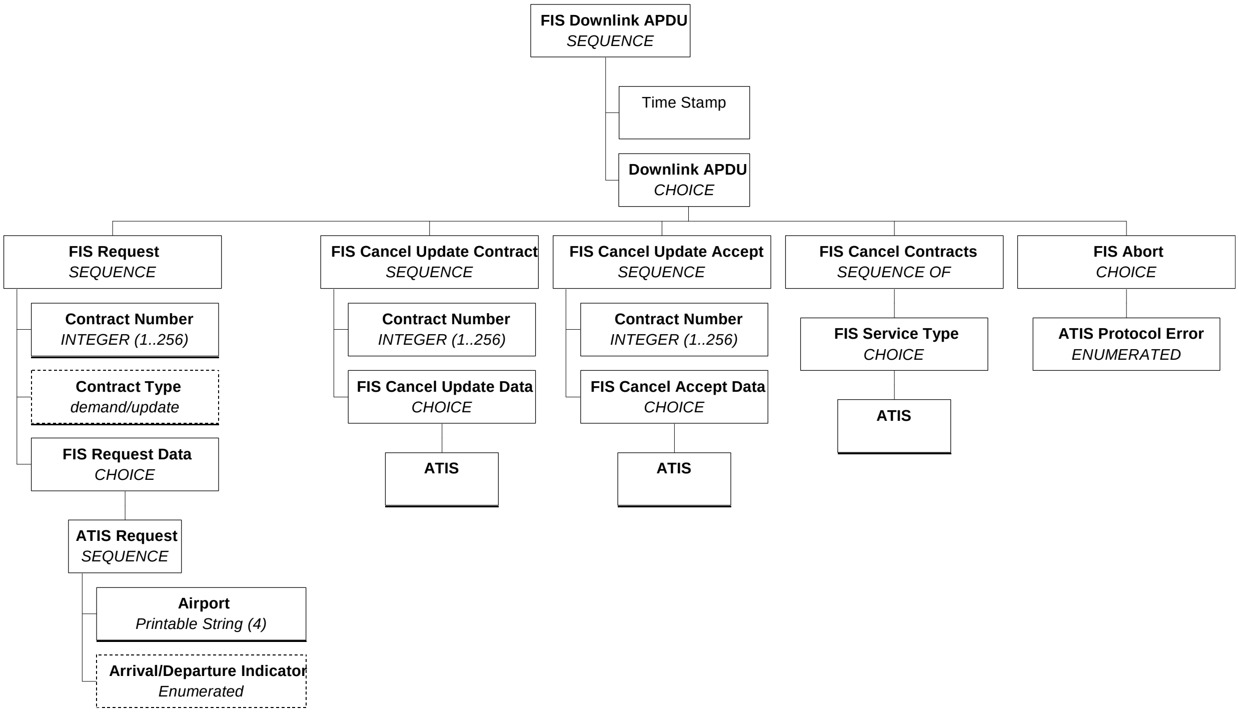


Figure 5.6-1. Downlink APDU Structure

5.6.2.3 Figure 5.6-2 represents the top-level APDU generated by the FIS-ground-ASE.

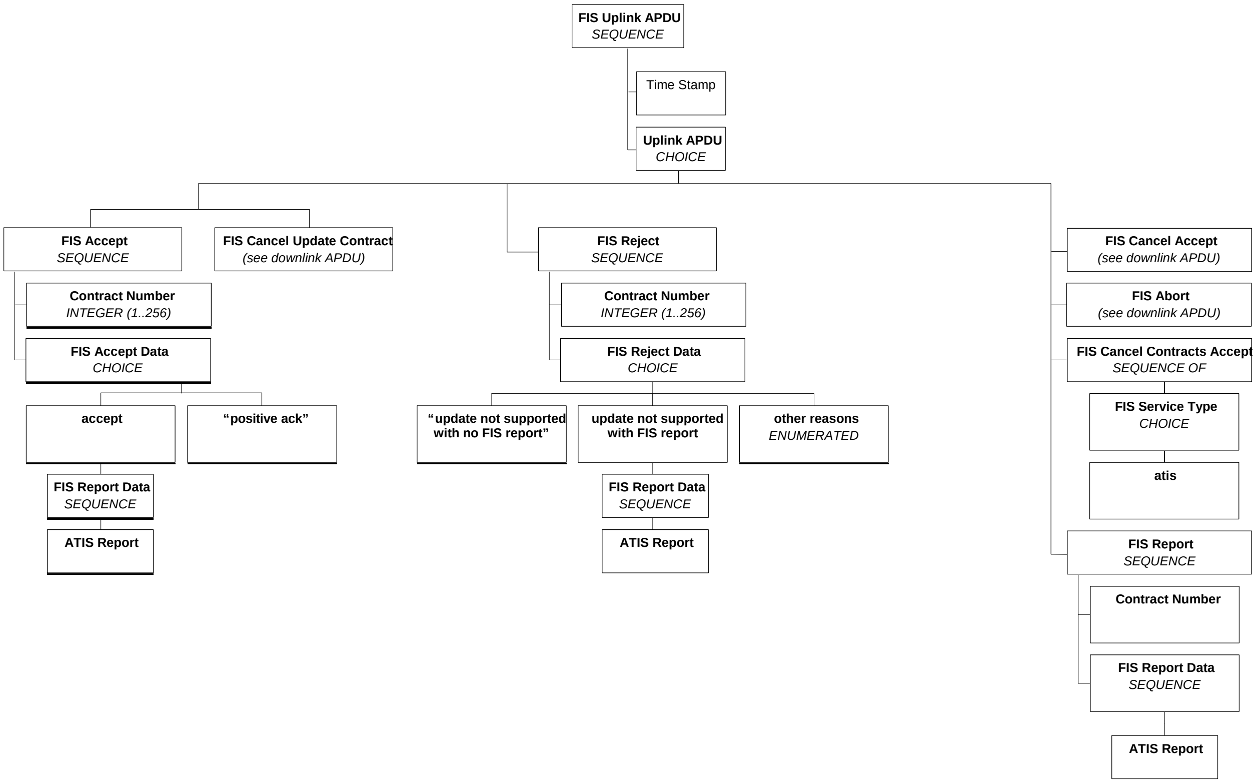


Figure 5.6-2. Uplink APDU Structure

5.6.2.4 Figure 5.6-3 represents the structure of an ATIS report uplinked to the aircraft.

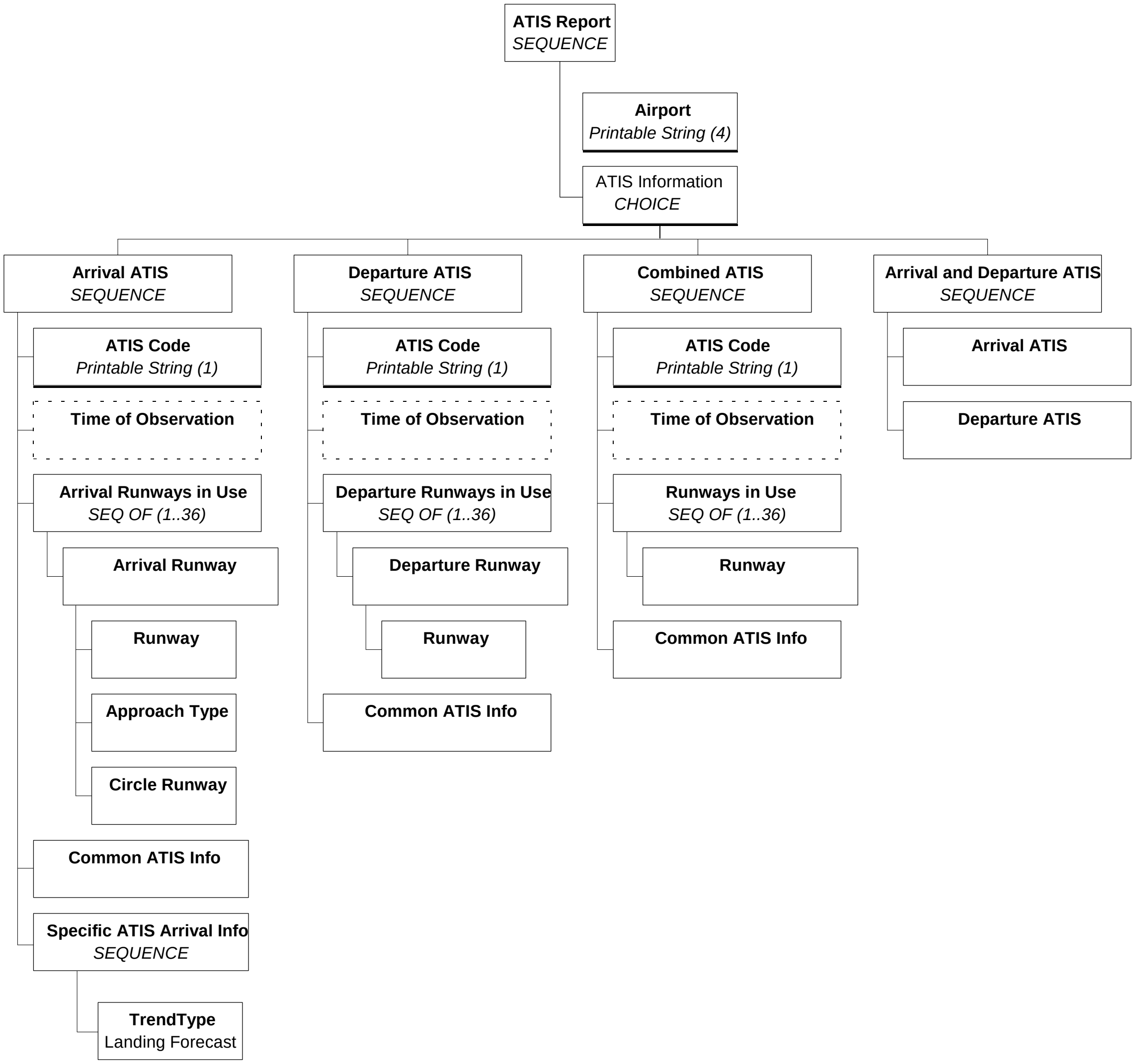


Figure 5.6-3. ATIS Report Structure

5.6.2.5 Figure 5.6-4 represents the structure of the ATIS fields present in all ATIS report (i.e. departure, arrival or combined).

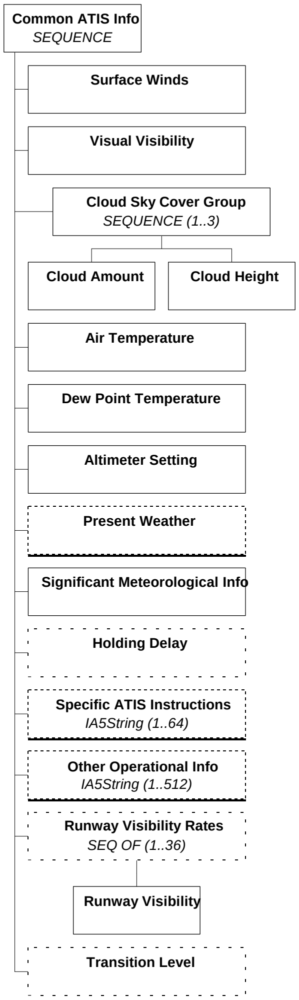


Figure 5.6-4. Common ATIS Structure

5.6.2.6 *ASN.1 Type Index*

5.6.2.7 Table 5.6-2 lists each ASN.1 type defined in the FIS SARPs in alphabetical order. In order to enable the definition to be found more easily, a cross reference to the ASN.1 section is given.

5.6.2.8 The third column lists those ASN.1 types that are used in the definition of the ASN.1 type in the first column, and the fourth column lists those ASN.1 types that use it. The third and fourth columns are therefore inverse references. The extensibility marker "…" indicates in the table that provision to define more values for a field has been made. The range and resolution are given where applicable in the fourth column.

Table 5.6-2.

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
Aerodrome	S4	IA5String	ATISReport ATISRequest	4 characters
AltimeterSetting	S4	PressureMeasure RunwayQFE	CommonATISInformation	
Approach	S4	ENUMERATED	ApproachType	Ils, localizer, ndb, vr, vordme, nonprecisiongps, precisiongps, dmearc, precisionapproachradar, asr, visual, rnav, chartedvisualcvap, lda, fms, lran, mls, ilsdme localizerbackcourse, localizerDME, vortac, tacan, ndbDme, vdf, sra, ...
ApproachType	S4	Approach IA5String (1..30)	ArrivalRunway CombinedRunway	
ArrivalAndDepartureATIS	S3	ArrivalATIS DepartureATIS	ATISInformation	
ArrivalATIS	S3	ArrivalRunway ATISDesignator CommonATIS Information SpecificATISArrival InfoTime	ArrivalAndDepartureATIS ATISInformation	
ArrivalDepartureIndicator	S4	ENUMERATED	ATISRequest	Arrival, departure, combined
ArrivalRunway	S4	Runway ApproachType RunwayId	ArrivalATIS RunwayType	
ATISDesignator	S4	IA5String	ArrivalATIS CombinedATIS DepartureATIS	1 character

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
ATISInformation	S3	ArrivalAndDeparture ATIS ArrivalATIS CombinedATIS DepartureATIS	ATISReport	
ATISReport	S3	Aerodrome ATISInformation	FISReportData	
ATISRequest	S3	Aerodrome ArrivalDeparture Indicator	FISRequestData	
BrakingAction	S4	BrakingAction Description	Runway	
BrakingActionDescription	S4	BrakingActionQuality IA5String (1..25)	BrakingAction	
BrakingActionQuality	S4	ENUMERATED	BrakingActionDescription	good, mediumToGood, medium, mediumToPoor, poor
Cloud	S4	CloudInformation SkyObscured NULL	CommonATISInformation	
CloudAmount	S4	ENUMERATED	CloudInformation	skyclear, few, scattered, broken, overcast,
CloudHeight	S4	INTEGER	CloudInformation	Unit: meters, Range: 0 to 3000, Res: "0 Or Unit: meters, Range: 3300..20100 Res: 300 Or Unit: feet Range: 0.10000, Res: 100 Or Unit: feet Range: 11000..60000 Res: 1000
CloudInformation	S4	CloudAmount CloudHeight CloudType	Cloud	
CloudType	S4	ENUMERATED	CloudInformation	Cumulonimbus, toweringCumulus
Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
ContractNumber	S4	INTEGER	FISAccept FISCancelUpdateAccept FISCancelUpdateContract FISReject	Range: 1 to 256

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
			FISReport FISRequest	
ContractType	S4	ENUMERATED	FISRequest	DemandContract, updateContract
CombinedRunway	S4	ApproachType Runway RunwayId	RunwayType	
CombinedATIS	S3	ATISDesignator CommonATIS Information RunwayType SpecificATISArrival Info Time	ATISInformation	
CommonATISInformation	S3	AltitudeSettingCloud IA5String PresentWeather SignificantMet Phenomena SurfaceWinds Temperature TransitionLevel Visibility	ArrivalATIS CombinedATIS DepartureATIS	
Date	S4	Day Month Year	DateTimeGroup	
DateTimeGroup	S4	Date HHMMSS	FISDownlinkAPDU FISUplinkAPDU	
Day	S4	INTEGER	Date	Unit: day Range: 1 to 31 Res: 1
DepartureATIS	S3	ATISDesignator CommonATIS Information DepartureRunway Time	ArrivalAndDepartureATIS ATISInformation	
DepartureRunway	S4	Runway	DepartureATIS RunwayType	
DescriptorQualifier	S4	ENUMERATED	PresentWX	Shallow, partial, patches, lowdrifting, blowing, shower, thunderstorm, freezing
DownlinkAPDU	S1	FISAbort FISCancelContracts FISCancelUpdate Accept FISCancelUpdate Contract FISRequest …	FISDownlinkAPDU	
FISAbort	S2	FISProtocolErrorDiag …	DownlinkAPDU UplinkAPDU	
FISAccept	S2	ContractNumber FISAcceptData	Uplink APDU	

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
FISAcceptData	S2	FISReportData NULL	FISAccept	
FISCancelAcceptData	S2	NULL …	FISCancelUpdateAccept	
FISCancelContracts	S2	FISServiceType	DownlinkAPDU	
FISCancelContractsAccept	S2	FISServiceType	UplinkAPDU	
FISCancelUpdateAccept	S2	ContractNumber FISCancelAcceptData	DownlinkAPDU UplinkAPDU	
FISCancelUpdateContract	S2	ContractNumber FISCancelUpdateData	DownlinkAPDU UplinkAPDU	
FISCancelUpdateData	S2	NULL …	FISCancelUpdateContract	
FISDownlinkAPDU	S1	DateTimeGroup DownlinkAPDU	-	
FISProtocolErrorDiag	S2	…	FISAbort	TimerExpiration, protocolError, sequenceError, decodingError, unrecoverable InternalError, invalidContract Numbzer, dialogueEndNot Supported, undefined, invalidQos Parameter
FISReject	S2	ContractNumber FISRejectData	UplinkAPDU	
FISRejectData	S2	FISRejectReason FISReportData NULL	FISReject	
FISRejectReason	S2	…	FISRejectData	CanNotComply, fISService Unavailable, errorinRequest, undefined
FISReport	S2	ContractNumber FISReportData	UplinkAPDU	
FISReportData	S2	ATISReport …	FISAcceptData FISRejectData FISReport	
FISRequest	S2	ContractNumber ContractType FISRequestData	DownlinkAPDU	
FISRequestData	S2	ATISRequest …	FISRequest	
FISServiceType	S2		FISCancelContracts FISCancelContractsAccept	atis
FISUplinkAPDU	S1	DateTimeGroup UplinkAPDU	-	
HHMMSS	S4	TimeMinutes TimeHours TimeSeconds	DateTimeGroup	

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
IntensityQualifier	S4	ENUMERATED	PresentWX	Light, moderate, heavy
Month	S4	INTEGER	Date	Unit: month Range: 1 to 12 Resolution: 1
Obscuration	S4	ENUMERATED	PresentWeatherType	Mist, fog, smoke, volcanicAsh, widespreadDust, sand, haze
Other**WeatherPhenomena**	S4	ENUMERATED	PresentWeatherType	dustsAndWhirls, squalls, funnelCloudTornadoWaterspout, sandstorm, duststorm
Precipitation	S4	ENUMERATED	PresentWeatherType	Drizzle, rain, snow, snowGrains, iceCrystals, icePellets, hail, smallHailAndOrSnowPellets, unknown Precipitation
PresentWeather	S4	NULL PresentWX	CommonATISInformation	
PresentWX	S4	DescriptionQualifier IntensityQualifer NULL PresentWeatherType	PresentWeather	
PresentWeatherType	S4	Precipitation Obscuration OtherWeather Phenomena	PresentWX	
PressureMeasure	S4	INTEGER	AltimeterSetting RunwayQFE	Unit: hPa Range: 500 to 1250 Res: 1 Or Unit: inches of Mercury Range: 22.00 to 32.00 Res: 0.01
Runway	S4	BrakingAction RunwayId RunwayArresting System RunwaySurface Conditions RVR	ArrivalRunway DepartureRunway CombinedRunway	
RunwayArrestingSystem	S4	RASLocation RASCondition	Runway	
RunwayDesignator	S4	INTEGER	RunwayId	Range: 1 to 36
RunwayId	S4	RunwayLetter RunwayDesignator	ArrivalRunway CombinedRunway	

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
			Runway RunwayQFE	
RunwayLetter	S4	ENUMERATED	RunwayId	leftLeft, left, leftCenter, center, rightCenter, right, rightRight
RunwayQFE	S4	PressureMeasure RunwayId	AltimeterSetting	
RunwaySurfaceConditions	S4	SurfaceConditions IA5String (1..256)	Runway	
RunwayType	S4	ArrivalRunway DepartureRunway CombinedRunway	CombinedATIS	
RunwayVisibility	S4	NULL RVRVisibility	RVR	
RASCondition	S4	ENUMERATED	RunwayArrestingSystem	up down unavailable
RASLocation	S4	ENUMERATED	RunwayArrestingSystem	arrivalEnd departureEnd
RVR	S4	RunwayVisibility RVRVisability	Runway	
RVR Visability	S4	INTEGER	RunwayVisibility RVR	Unit: meter, Range: 0 to 800, Res: 25+ Or Unit: meters, Range: 900..1500, Res: 100 Or Unit: feet Range: 0..1000, Res: 100 Or Unit: feet Range: 1200..3000 Res: 200 Or Unit: feet Range: 3500..6000 Res: 500
SignificantMetPhenomena	S4	IA5String (1..128)	CommonATISInformation	
SkyObscured	S4	NULL VerticalVisibility	Cloud	
SpecificATISArrivalInfo	S3	IA5String(1..256)	ArrivalATIS CombinedATIS	
SurfaceConditions	S4	ENUMERATED	RunwaySurfaceConditions	damp,wet,water

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
				Patches, flooded,wetSnow, drySnow,snow, slush,ice
SurfaceWinds	S4	NULL surfaceWD	CommonATISInformation	
SurfaceWD	S4	SurfaceWindDirection SurfaceWindSpeed SurfaceWind Variations	SurfaceWind	
SurfaceWindDirection	S4	INTEGER	SurfaceWD SurfaceWindVariations	Unit: degree Range: 1 to 36, Res: 10
SurfaceWindSpeed	S4	INTEGER	SurfaceWD SurfaceWindVariations	Unit: kmperhour Range: 0 to 500, Res: 1 Or Unit: knots Range: 0 to 200, Res: 1
Temperature	S4	INTEGER, NULL	CommonATISInformation	Unit: °Celcius Range: -80..60, Res: 1 Or notavailable
Time	S4	TimeHours TimeMinutes	ArrivalATIS CombinedATIS DepartureATIS	
TimeHours	S4	INTEGER	HHMMSS Time	Unit: Hour Range: 0 to 23, Res: 1
TimeMinutes	S4	INTEGER	HHMMSS Time	Unit: minute Range: 0 to 59, Res: 1
TimeSeconds	S4	INTEGER	HHMMSS	Unit: second Range: 0 to 59, Res: 1
TransitionLevel	S4	INTEGER	CommonATISInformation	Unit: meters, Range: 1000..2000, Res: 10 Or Unit: feet, Range: 3000..60000, Res: 100
UplinkAPDU	S1	FISAbort FISAccept FISCancelContracts Accept FISCancelUpdate Accept FISCancelUpdate Contract FISReject FISReport ...	FISUplinkAPDU	

Type	*ASN.1 Section*	*Type used by this type*	*Types using this type*	*Unit, Range and Resolution*
VerticalVisibility	S4	INTEGER NULL	SkyObscured	Unit: meters Range: 0..3000, Res: 30 Or Unit: feet Range: 0..10000, Res: 100 Or Not available
VisibilityStatuteMiles	S4	INTEGER NULL	Visibility	Unit: 1/16th or 1/8th or 1/4th 1 or 5 statute mile or more than 50
Visibility	S4	INTEGER VisibilityStatuteMiles	CommonATISInformation	Unit: meter Range: 0 to 500, Res: 50 Or Unit: meters Range: 600..4900 Res: 100 Or Unit: km Range: 5 to 10, Res: 1 Or Cf.VisibilityStatute Miles
Year		-	Date	Unit: year Range: 1996 to 2095 Resolution: 1
...(extensibility marker)	-	-	DownlinkAPDU FISAbort FISCancelAcceptData FISCancelUpdateData FISProtocolErrorDiag FISRejectReason FISReportData FISRequestData FISServiceType UplinkAPDU	

5.6.3 ASN.1 Glossary

5.6.3.1 Two sets of data types are defined in the FIS ASN.1 description. The first set (FIS level defined in ASN.1 sections S1 and S2) contains information common to all type of FIS services. The second set (ATIS level defined in ASN.1 sections S3 and S4) contains information related to a specific DFIS service, here the ATIS service.

5.6.3.2 *FIS Protocol Data*

5.6.3.2.1 These data are used to co-ordinate the processing of remote FIS ASEs. They are Protocol Control Information which do not carry operational data but are used for protocol processing purposes. They are used:

a) to identify the type of the APDU transmitted on the line (e.g. FISDemandContract Downlink APDU or FISAbort Uplink APDU),

b) to identify the type of FIS service the contents of the APDU is related to (e.g. ATIS, METAR, etc.), and

c) to identify the contract the APDU is related to.

5.6.3.2.2 The following data are used as the FIS message variables, or component of the variables, and are shown here in the alphabetic order:

ContractNumber	Identifies the contract to which the APDU is related to.
DownlinkAPDU	Contains the identification and the actual contents of the downlink FIS messages.
FISAbort	Specifies the reason of an abort caused by the FIS ASE.
FISAccept	Identifies the FIS contract being accepted and optionally a FIS report.
FISAcceptData	Contains data sent back with a contract acceptation message.
FISCancelContracts	Identifies the types of FIS contracts being requested to be cancelled.
FISCancelContractsAccept	Identifies the types of FIS contracts the cancellation of which is accepted.
FISCancelAcceptData	Contains data sent back with a contract cancellation message.
FISCancelUpdateAccept	Identifies the FIS update contract being cancelled.
FISCancelUpdateContract	Identifies the FIS update contract being requested to be cancelled.
FISCancelUpdateData	Contains data sent back with the request of a contract cancellation.
FISDownlinkAPDU	Contains any FIS downlink message with a time stamp.
FISProtocolErrorDiag	Specifies the reason of the abort by either of the FIS ASEs.
FISReject	Identifies the FIS contract being rejected with a reason for rejection.
FISRejectData	Contains data sent back with a contract rejection.
FISRejectReason	Identifies the reason of the contract rejection.
FISReport	Identifies the FIS contract and contains the fields of the FIS report.
FISReportData	Contains the fields of a FIS report.
FISRequest	Identifies the FIS contract being established and contains the parameters of the contract.
FISRequestData	Contains the parameters of the FIS contract being established.
FISServiceType	Identifies the type of a FIS contract.
FISUplinkAPDU	Contains any uplink FIS message with a time stamp.
UplinkAPDU	Contains the identification and the actual contents of the uplink FIS messages.

5.6.3.2.3 **ATIS Protocol Data**

5.6.3.2.4 These data are used to co-ordinate the processing between FIS ASEs providing the ATIS service. They are used to convey the operational information likely to be generated by ATIS users.

5.6.3.2.5 Some types are common to several data link applications. However, even if the semantic is the same, the range and resolution required by each application may be specific. Therefore it has been decided to specify stand-alone ASN.1 descriptions completely independent from the others.

5.6.3.2.6 The following data are used as the ATIS message variables, or component of the variables, and are shown here in the alphabetic order:

Aerodrome Indicates the aerodrome for the ATIS information.

AltimeterSetting Indicates the QNH, and optionally the QFE for specified runway(s) or for the aerodrome.

Approach Specifies names approach types: ILS, ILS/DME, Localizer, Localizer Back course, Localiver DME, NDB, NDB/DME, VOR, VOR/DME, VORTAC, TACAN, Precision GPS, Non Precision GPS, DME/ARC, Precision approach Radar, Airport Surveillance Radar, Visual, Charted Visual Approach Procedure, LDA FMS, Loran, NAV, MLS, SRA or VDF.

ApproachType Indicates the type of approach to be expected, as a named approach or a character string .

ArrivalAndDepartureATIS Provides both arrival and departure ATIS fields.

ArrivalATIS Provides arrival ATIS fields and the time at which the meteorological obsrvation was made.

ArrivalDepartureIndicator Indicates which type of ATIS information is provided (i.e. arrival, departure or combined).

ArrivalRunway Specifies a landing runway. May be further described as a runway which is circled to for landing.

ATISDesignator Provides a letter identifying the ATIS iteration.

ATISInformation Contains the fields of an arrival, departure, combined or both arrival and departure ATIS.

ATISReport Provides the airport identifier and the ATIS related to it.

ATISRequest Identifies the airport identifier and the type of ATIS requested.

BrakingAction Provides braking action for full runway or by thirds of the runway in case of contaminated runway conditions.

BrakingAction Description Specifies the braking action as a standardised value, as well as asociated explanatory text.

BrakingActionQuality Indicates the braking action in case of contamined runway conditions as Good, Medium to Good, Medium, Medium to Poor and Poor.

Cloud Provides Cloud Information of any cloud below 1500 m (5000 ft) or below the highest minimum sector level whichever is greater, an indication that the sky is obscured, or CAVOK. CAVOK replaced Cloud, visibility and Present Weather. When the sky is obscured, the *Vertical*

Visibility is provided.

CloudAmount Specifies cloud amount: sky clear (SKC), few (FEW), scattered (SCT), broken (BKN), or overcast.

CloudHeight Indicates the height of the base of the clouds above ground level in meters or feet.

CloudInformation Provides cloud information: amount, height and type.

CloudType Identified only for cumulonimbus or towering cumulus.

CombinedATIS Provides the field of a combined ATIS and the time at which the meteorological observation was made.

CombinedRunway Specifies a landing and departure runway. May be further described - for arrival only - as a runway which is circled to for landing.

CommonATISInformation Provides the ATIS fields common to both departure and arrival. *Dew point temperature* indicates the outside dew-point temperature représentative of the runway(s). *Air temperature* indicates the outside air temperature representative of the runway(s). *Holding Delay* indicates an unplanned delay prior to the initiation of the next phase of flight. *Specific ATIS Instructions* gives instruction to the pilot to acknowledge the receipt of the ATIS message upon initial contact. *Other operational Info* contains information not covered by the other available information. Other ATIS fields are: *SurfaceWind*, *Visibility*, *Cloud*, *Altimeter Setting*, *Present Weather*, *Significant Met Phenomena* and *Transition Level*.

ContractNumber Identifying number of each contract.

ContractType Distinguishes between the types of ATIS provided (demand or contract).

Date Provides date as year, month and day.

DateTimeGroup Provides date and time as hours, minutes and seconds.

Day Specifies the day of the month.

DepartureATIS Provides the fields of a departure ATIS and the time at which the meteorological observation was made.

DepartureRunway Identifies a departure runway.

DescriptorQualifier Describes a present weather phenomenon as: shallow, partial, patches, low drifting, blowing shoer(s), thunderstorm, or freezing.

HHMMSS Provides the time as hours, minutes and seconds.

IntensityQualifier Describes a present weather phenomenon as: light, moderate, or heavy,

Month Specifies the month of the year.

Obscuration see *PresentWeatherType*,

OtherWeatherPhenomena see *PresentWeatherType*,

Precipitation see *PresentWeatherType*,

PresentWeather Gives the present weather (*PresentWX*) occuring at or near the aerodrome and may indicate NOSIG. NOSIG is a contraction which indicates that the reported presentweather is not expected to change or pass a specific threshold value.

PresentWeatherType Identifies the present weather as precipitation (i.e. drizzle, rain, snow, snow grains, ice crystals, ice pellets, hail or small hail/or snow pellets), *Obscuration* (i.e. mist, fog, smoke, volcanic ash, widespread dust, sand or haze) or *Other Weather Phenomena* (i.e. dust/sand whirls, squall, funnel clouds (tornado/waterspout), sandstorm or duststorm).

PresentWX The present weather phenomena are selected from the *PresentWeatherType* which is described by an *IntensityQualifier*,

	ProximityQualifier or *DescriptorQualifier*, *ProximityQualifier* is a boolean indication “in/not in the vicinity”.
PressureMeasure	Provides the pressure expressed in hecto pascal or inches of mercury.
Runway	Provides the *Runway Surface Conditions*, the *Braking Action* and the *Arresting System* of a runway.
RunwayArrestingSystem	Indicates when an arresting system constitutes a potential hazard. Expressed per runway as *RAS location* and *RAS condition*.
RunwayDesignator	Identifies the runway by a two digit number.
RunwayId	Provides the identity of a runway using a Runway Designator and optionally an addition Runway Letter.
RunwayLetter	Identifies the runway expressed as leftfet, left, lefcenter, centre, rightcenter, right or rightright.
RunwayQFE	Provides the qFE measured for a runway.
RunwaySurfaceConditions	Describes the *Surface Conditions* followed by the text to further explain details of the conditions, such as depth.
RunwayType	Identifies a runway as an arrival, departure or combined runway.
RunwayVisibility and RVRVisibility	The range (in meters or feet) over which the pilot of an aircraft on the center line of a runway can see the runway surface markings or the lights delineating the runway or identifying its center line. Can indicates that the meausre is inoperative.
RASCondition	Describes the runway arresting system condition as either up, down or unavailable.
RASLocation	Describes the runway arresting system location as being at the arrival, departure, or both ends of the runway.
RVR	The *Runway Visibility* values representative of touchdown zone, mid-point and stop-end which are included when available, or reported as inopertive or not reported, when applicable. Otherwise, only the value representative of the touchdown zone is provided. RVR may also include, if required, the RVR Variation Qualifier.
SignificantMetPhenomena	Gives information on significant meteorological phenomena specific to Approach area MET, take-off area MET, and/or Climb-out area MET.
SkyObscured	See *Cloud.*
SpecificATISArrival Info	Contains ATIS fields specific to the arrival ATIS, as the trend type landing forecast which is a concise statement of the expevcted trend of the meteorological conditions at the aerodrome.
SurfaceConditions	Describes the runway conditions as being damp, wet, water patches, flooded, snow, wet snow, dry snow, slush, or ice.
SurfaceWind	Contains *Surface WD* or the indicator “calm”.
SurfaceWindDirection	Indicated in degrees magnetics.
SurfaceWindSpeed	Indicated in kilometre per hour mile per hour or knots.
SurfaceWindVariations	Provides significant variations of wind direction and wind speed in terms of two extreme directions and maximum wind speed (gusts). Expressed as *Surface Wind Speed* and two of *Surface Wind Direction*.
SurfaceW	Indicates *Surface Wind Direction*, *Surface Wind Speed* and, when appropriate, *Surface Wind Variations*.
Temperature	Indicates the temperature in degrees Celsius, or as “not available”.
Time	Provides the time in hours and minutes.
TimeHours	Specifies time in hours of a day.

TimeMinutes Specifies time in minutes of an hour.
TimeSeconds Specifies time in seconds of a minute.
TransitionLevel Indicates a change from the published transition level. Expressed as flight level in hundreds of feet or tens of meters.
VerticalVisibility Defined as the vertical visual range (in meters of feet) into an obscuring medium. Included instead of Cloud amount, height and type when the sky is obscured.
VisibilityStatuteMiles Specifies a distance in statute miles.
Visibility Specifies a distance in meters, kilometres or statute miles.
Year Specifies the year between 1996 and 2095.

5.7 Example of Operational Scenarios

5.7.1 Introduction

5.7.1.1 This section contains a set of example scenarios of use. The purpose of this section is to demonstrate some scenarios that are theoretically possible using FIS. It is not meant to indicate what is required from an operational point of view.

5.7.1.2 The aircraft is equipped with CM, ADS, CPDLC and ATIS applications.

5.7.2 Before take-off

a) the pilot initiates the CM-Logon process with the ground CM covering the departure airport. The CM's response includes the address information for the ground CPDLC and ATIS applications;

b) due to the potential for last minute weather changes, the pilot initiates a request for an ATIS Update Contract for the departure airport. The ground FIS system accepts the contract and sends the current ATIS to the aircraft;

c) an indication alerts the flight crew to an updated ATIS. This ATIS indicates that the forecasted weather has arrived in the airport environment, and warns the crew of the approaching thunderstorms; and

d) the rain has arrived at the airport now, and the flight crew receives another update from the ATIS contract. Although conditions are deteriorating, the RVR is acceptable for departure.

5.7.3 After take-off

a) the aircraft is now leaving the TMA environment and the pilot elects to cancel the ATIS contract for the departure airport;

b) the pilot initiates the CM-Logon process with the ground CM covering the destination airport;

c) the pilot initiates an ATIS Demand Contract for the destination airport. He informs the passengers about the weather conditions at the arrival airport;

d) time to time, the pilot initiates an ATIS Demand Contract for the arrival airport; and

e) one hour before arrival time, the pilot establishes an ATIS Update Contract for the destination airport. The ATIS information indicates that conditions at the airport are conducive to an approach to the alternate arrival runway which provides a smoother more efficient profile for the aircraft.

5.7.4 After landing

a) the pilot/avionics requests the cancellation of all FIS contracts still pending.

5.8 Example Encoding

5.8.1 In the following examples of APDU encoding, the tables present for each ASN.1 field the actual values being encoded and the result of the PER encoding[3]. To make it easier to read the binary view of the data, a period (.) is used to mark octet boundaries ; and an 'x' represents a zero-bit used to pad the final octet to an octet boundary.

[3] The PER encoding presented in this section has been provided “by hand” and should be checked with a reference PER compiler.

5.8.2 FIS Downlink APDU [FISRequest]

Element	*Type*	*Value*	*Encoding*	*Comments*
FISDownlinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
timeSeconds	INTEGER (0..59)	0	0	
FisDownlinkAPDU	CHOICE	FISRequest	0	No extension used The CHOICE [0] is selected
FISRequest SEQUENCE	SEQUENCE		0	ContractType not present
ContractNumber	INTEGER (1..256)	1	0	
ContractType		DemandContract		
FISRequestData	CHOICE	ATISRequest	0	No extension used The CHOICE [0] is selected
ATISRequest	SEQUENCE		0	ArrivalDepartureIndicator no present
AirportID	PrintableString (SIZE(4))	"LFBO"	0100110. 0010001 1.001000 01.00100 111.1xxx xxxx	
ArrivalDepartureIndicator		Arrival		

Hexadecimal view (11 octets)

```
01 6b 5b a0 00 00 26 23 21 27 80
```

5.8.3 FIS Downlink APDU [FISCancelUpdateContract]

Element	*Type*	*Value*	*Encoding*	*Comments*
FISDownlinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
timeSeconds	INTEGER (0..59)	0	0	
FISDownlinkAPDU	CHOICE	FISCancelUpdate Contract	1	No extension used The CHOICE [1] is selected
FISCancelUpdateContract	SEQUENCE			
ContractNumber	INTEGER (1..256)	1	0	
FISCancelUpdateData	CHOICE	atis	0x	No extension used The CHOICE [0] is selected
atis	NULL			

Hexadecimal view (6 octets): 01 6b 5b a0 08 00

5.8.4 **FIS Downlink APDU [FISCancelUpdateAccept]**

Element	*Type*	*Value*	*Encoding*	*Comments*
FISDownlinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
timeSeconds	INTEGER (0..59)	0	0	
FISDownlinkAPDU	CHOICE	FISCancelUpdate Accept	10	No extension used The CHOICE [2] is selected
FISCancelUpdateAccept	SEQUENCE			
ContractNumber	INTEGER (1..256)	1	0	
FISCancelAcceptData	CHOICE	atis	0x	No extension used The CHOICE [0] is selected
atis	NULL			

Hexadecimal view (6 octets): 01 6b 5b a0 10 00

5.8.5 **FIS Downlink APDU [FISCancelContracts]**

Element	*Type*	*Value*	*Encoding*	*Comments*
FISDownlinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
timeSeconds	INTEGER (0..59)	0	0	
FisDownlinkAPDU	CHOICE	FISCancelContracts	11	No extension used The CHOICE [3] is selected
FISCancelContracts	SEQUENCE OF		0	Number of element is less than 127 1 element in the SEQUENCE OF
FISServiceType	ENUMERATED (0)	atis (0)	0xx	No extension used

Hexadecimal view (6 octets): 01 6b 5b a0 18 08

5.8.6 **FIS Downlink APDU [FISAbort]**

Element	*Type*	*Value*	*Encoding*	*Comments*
FISDownlinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
TimeSeconds	INTEGER (0..59)	0	0	
FISDownlinkAPDU	CHOICE	FISAbort	100	No extension used The CHOICE [4] is selected
FISAbort	CHOICE	Atis	0	No extension used The CHOICE [0] is selected
Atis	ENUMERATED (0..8)	ProtocolError (1)	0 0.001xxx xx	No extension used

Hexadecimal view (6 octets): 01 6b 5b a0 20 20

5.8.7 FIS Uplink APDU [FISAccept(positive-acknowledgement)]

Element	*Type*	*Value*	*Encoding*	*Comments*
FISUplinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
TimeSeconds	INTEGER (0..59)	0	0	
FisUplinkAPDU	CHOICE	FISAccept	0	No extension used The CHOICE [0] is selected
FISAccept	SEQUENCE			
ContractNumber	INTEGER (1..256)	1	0	
FISAcceptData	CHOICE		1x	The CHOICE[1] is selected

Hexadecimal view (6 octets): 01 6b 5b a0 00 02

5.8.8 FIS Uplink APDU [FISAccept(positive-acknowledgement)]

Element	*Type*	*Value*	*Encoding*	*Comments*
FISUplinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
TimeSeconds	INTEGER (0..59)	0	0	
FisUplinkAPDU	CHOICE	FISReject	1	No extension used The CHOICE [1] is selected
FISReject	SEQUENCE			
ContractNumber	INTEGER (1..256)	1	0	
FISRejectData	CHOICE	FISRejectReason	10	The CHOICE[2] is selected
FISRejectReason	ENUMERATED (0..3)	ErrorInRequest (3)	0 11xxxxx	No extension used

Hexadecimal view (7 octets): 01 6b 5b a0 08 02 c0

5.8.9 FIS Uplink APDU [FISReport]

Element	*Type*	*Value*	*Encoding*	*Comments*
FISUplinkAPDU	SEQUENCE			
time	SEQUENCE			
Date	SEQUENCE			
Year	INTEGER (1996..2095)	1996	0	
Month	INTEGER (1..12)	12	1.011	
Day	INTEGER (1..31)	12	1011	
time	SEQUENCE			
TimeHours	INTEGER (0..23)	11	1011	
TimeMinutes	INTEGER (0..59)	29	11.101	
timeSeconds	INTEGER (0..59)	0	0	
fisUplinkAPDU	CHOICE	FISReport	0	No extension used The CHOICE [0] is selected
fISReport	SEQUENCE			
ContractNumber	INTEGER (1..256)	1	0	
fISReportData	CHOICE			No extension used The CHOICE [0] is selected
Atis	SEQUENCE			
AerodromeID	IA5String (SIZE(4))	“LFBO”	01.001100 01.000110 01.000010 01.001111	
aTISInformation	CHOICE		0	The CHOICE [0] is selected
ArrivalATIS	SEQUENCE		1	ATISTimeOfObservation is present
ATISDesignator	IA5String (SIZE(1))	“A”	10000.01	
ATISTimeOfObservation	SEQUENCE			
TimeHours	INTEGER (0..23)	14	1110	
TimeMinutes	INTEGER (0..59)	13	0.01101	
ArrivalRunwayinUse	SEQUENCE OF (1..36)		0	1 element
ArrivalRunway	SEQUENCE		10	ApproachType is present
Runway	SEQUENCE		10	BrakingAction is present
RunwayId	SEQUENCE		1	RunwayLetter is present
RunwayDesignator	INTEGER (1..36)	3	1	
RunwayLetter	ENUMERATED (0..6)	left (1)	1	
BrakingAction	SEQUENCE		1	BrakingActionFull is present

Element	*Type*	*Value*	*Encoding*	*Comments*
BrakingActionFull	SEQUENCE			
BrakingActionQuality	ENUMERATED (0..4)	Good (0)	0	
BrakingActionQualifier	IA5String (1..25)	"AA"	00001. 01000001. 01000001.	Length=2
ApproachType	SEQUENCE		0	
Approach	ENUMERATED (0..24)	Ils (0)	0	
CommonATISInformation	SEQUENCE		0	No extension used no optional field present
SurfaceWinds	CHOICE		0	CHOICE [0] is selected
CalmIndicator	NULL	nil		
Cloud	CHOICE		0	CHOICE [2] is selected
CAVOK	NULL	nil		
AirTemperature	CHOICE		1	CHOICE [1] is selected
Temperature	INTEGER (-80..60)	10	101101	
DewPointTemperature	CHOICE		1	CHOICE [1] is selected
Temperature	INTEGER (-80..60)	0	10100	
AltimeterSetting	SEQUENCE		0	No optional field is present
QNH	CHOICE		0	CHOICE [0] is selected
Hpa	INTEGER (500..1250)	500	0	
ArrivalATISInfo	SEQUENCE		0. 0 xxxxxxx	No extension used No optional field present padding bits

Hexadecimal view (27 octets)

01 6b 5b a0 00 01 31 19 09 3c 50 58 68 12 84 68 00 08 78 3e 85 a2 80 8c 61 40 00

5.9 Identified Problems

5.9.1 Several defects identified in the ICAO Edition 2.2 SARPs were not corrected yet. These defects do not jeopardize the interoperability capability of the FIS ASEs and are not considered as safety-critical.

5.9.2 This section lists the defects known at the time of the publication of the CAMAL and describes the behavior of the ASEs when they occur.

5.9.3 Except for the first problem listed below, all the resolutions of the associated Proposed Defect Reports (PDRs) may be implemented upon an unilateral choice since the resolution do not impact the interoperability with the peer.

5.9.4 The reader is invited to refer to the PDRs to get more details.

5.9.5 Positive Acknowledgment on a negative FIS-contract-update response

5.9.5.1 The ICAO *Manual of Air Traffic Services Data Link Applications* (Doc 9694) applications states that, upon receipt of an update contract indication, in case the FIS ground system does not support the contract mode and the requested FIS information is not immediately available, the FIS ground system shall be able to send a positive acknowledgment first and later on send the update contract reject along with the FIS information.

5.9.5.2 The current SARPs do not allow to send subsequently a positive acknowledgment and an update contract rejection. It imposes the ground system to reject the update contract immediately.

5.9.6 Simultaneous air and ground contract cancellation

5.9.6.1 The collision case where the FIS-air-user invokes the FIS-cancel-contracts service at the same time the FIS-ground-user invokes the FIS-cancel-update-contract on a particular contract is not correctly handled by the FIS protocol.

5.9.6.2 This collision situation causes the abort of the underlying dialogue and the automatic abort of all pending contracts, even those not associated with the FIS-cancel-contracts service.

5.9.6.3 The impact is very small in Package-1 since there is only one type of contract defined (ATIS) and the FIS-cancel-contracts therefore causes the release of all pending contracts. Therefore, the only difference is that the pending contracts are aborted rather than properly canceled.

5.9.7 FIS-abort-indication Reason parameter

5.9.7.1 The definition of the FIS-provider-abort *Reason* parameter missed three valid values "cannot establish contact with the peer", "contact refused by the peer" and "communication failure".

5.9.7.2 This problem of incompatibility between the protocol specification and the service definition will be fixed in the next version of the SARPs.

5.9.8 T-inactivity timer management

5.9.8.1 The protocol specification omits to request the air UC module to start the t-inactivity timer upon receipt of [FISCancelUpdateAccept] APDU. As a result, the dialogue is maintained open.

5.9.8.2 The protocol specification request the air UC module to start the t-inactivity timer on receipt of a [FISCancelUpdateContract] APDU. In the collision situation, this is too early, the timer should be started later, after the receipt of the [FIsCancelUpdateAccept] APDU. As a result, the dialogue is closed sooner.

5.9.9 Invalid list of APDU allowed in reception of a D-START confirmation

5.9.10 The protocol specification does not allow the air LI module to receive a [FISCancelUpdateContract] APDU in the D-START *User Data* parameter. This should be allowed when the ground cancellation occurs during the contract establishment phase. As a result, the underlying dialogue and all pending FIS contracts are all aborted.

5.9.11 Invalid state change

5.9.11.1 The protocol specification should request both sides to stay in the UC-CANCEL state upon receipt of a [FISCancelUpdateContract] APDU in the UC-CANCEL state (FIS-cancel-update service collision). The current specification requests incorrectly the ground UC module to move to the IDLE state. As a result, the underlying dialogue and all pending FIS contracts are all aborted.

— — — — — — — —

6. ATS MESSAGE HANDLING SERVICES (ATSMHS)

6.1 Overview

6.1.1 Introduction

6.1.1.1 *Purpose of the SARPs*

6.1.1.1.1 The ATN (Aeronautical Telecommunication Network) SARPs (Standards and Recommended Practices) for ATS (Air Traffic Services) Message Handling Services (ATSMHS) define two applications which allow ATS Messages to be exchanged between service users. These two ATS Message Handling Services are generic messaging services over the ATN Internet:

a) the ATS Message Service, which is a store-and-forward messaging service over the ATN Internet; and

b) the ATN Pass-Through Service, which is a transmission facility over the ATN Internet for AFTN (Aeronautical Fixed Telecommunication Network) messages.

6.1.1.1.2 Three categories of ATN End Systems are defined for the support of the ATS Message Service. They are the ATS Message Server, the ATS Message User Agent and the AFTN/AMHS Gateway (Aeronautical Fixed Telecommunication Network/ATS Message Handling System). Together, they provide connectivity between users at ATN End Systems and users at AFTN Stations in three different end-to-end configurations:

a) from an AFTN Station to another AFTN Station over the ATN;

b) from an AFTN Station to an ATN End System, and vice-versa;

c) from an ATN End System to another ATN End System.

6.1.1.1.3 A single category of ATN End System is defined for the support of the ATN Pass-Through Service. It is the AFTN/ATN Type A Gateway. The use of two peer AFTN/ATN Type A Gateways interconnected over the ATN Internet provides an end-to-end connectivity between two users at AFTN Stations over the ATN Internet.

6.1.1.1.4 These two aspects are depicted in Figure 6.1-1.

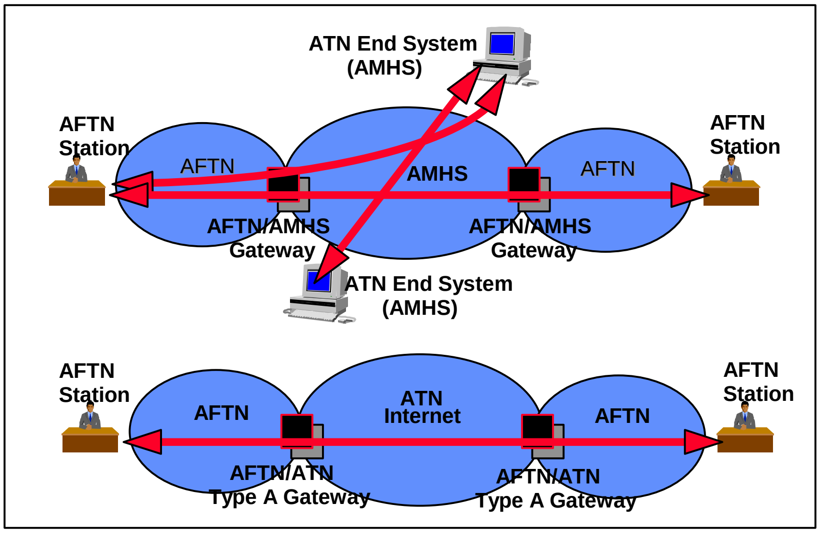


Figure 6.1-1. ATSMHS Traffic flows

6.1.1.1.5 The implementation of the ATS Message Service is mandatory for conformance with the SARPs. However, as a matter of organisations' policy, interim conformance may be achieved with the implementation of the ATN Pass-Through Service. The choice to implement the ATN Pass-Through Service as an interim solution does not replace the requirement to implement the ATS Message Service at the earliest possible date.

6.1.1.1.6 The choice to implement the ATN Pass-Through Service also implies the requirement to provide the interoperability facilities to the ATS Message Service implementations. Such facilities between the ATS Message service and the ATN Pass-Through Service are a local implementation matter, provided that the behaviour exhibited externally to the facility is identical to that of an AFTN/AMHS Gateway and of an AFTN/ATN Type A Gateway, respectively.

6.1.1.2 *History*

6.1.1.2.1 The Aeronautical Fixed Service (AFS) Systems Planning for Data Interchange Panel (ASPP), at its ASPP/3 meeting, included, among other things, the following work items in its future work programme:

a) Monitoring of implementation of AFTN and development of solutions to related problems, including the changes required to support the mixed AFTN/ATN environment; and

b) implications of the proposed ATN concept.

6.1.1.2.2 These work items resulted in the production by the ASPP Working Group of a draft Manual on ATS Message Handling over the ATN. This draft Manual included material concerning ATS Message Handling and the related AFTN/ATN Gateway. Subsequent to the ASPP dissolution, and to the incorporation of the ASPP terms of reference in those of the Aeronautical Telecommunication Panel (ATNP), this material was presented to ATNP/1.

6.1.1.2.3 The ATNP/1 meeting concluded that this material was sufficiently mature and recommended that the draft Manual on ATS message handling over the ATN be published as an ICAO Manual. Furthermore, this draft Manual was recognized as the basis for the future development of the Draft SARPs for ATS Message Handling Services.

6.1.1.2.4 The draft Manual included the specification of two ATS Message Protocol Stacks called Type A and Type B. These two protocol stacks have led to the definition of the applications comprised in ATS Message Handling Services, the ATS Message Service and the ATN Pass-Through Service, which implement the ATS Message Protocol Stack Type B and the ATS Message Protocol Stack Type A, respectively.

6.1.1.2.5 At the ATNP/2 meeting, the Panel recognized that the technical specification included in the draft Manual was superseded by the material included in the ATN SARPs, under the title ATSMHS Message Handling Services (Chapter 3.1 of Sub-Volume III of the *Manual of Technical Provisions for the ATN* (Doc 9705). It was later determined that the initial draft Manual, whose publication had been withheld pending results of ATNP/2, should not be published by ICAO.

6.1.1.3 *Scope, purpose and structure*

6.1.1.3.1 This chapter of Part III provides guidance material for implementers, service providers and users of ATS Message Handling Services.

6.1.1.3.2 It has been developed as a companion document to the ATSMHS SARPs. It may be read alongside the SARPs, and for this purpose the structure of this chapter, at a high level, has been aligned on the structure of the SARPs. This means that, e.g. section 6.2 of this chapter provides guidance on the subjects addressed in section 3.1.2 of the SARPs (where 3.1 identifies the ATSMHS SARPs in the overall ATN SARPs).

6.1.2 **Application functionalities**

6.1.2.1 *ATS Message Service Overview*

6.1.2.1.1 Two levels of service are defined within the ATS Message Service:

a) the Basic ATS Message Service.

b) the Extended ATS Message Service.

6.1.2.1.2 The Basic ATS Message Service meets the basic requirements of the first version of the Message Handling Systems (MHS) Profiles published by the International Organization for Standardization (ISO) as International Standardized Profiles (ISPs), and it incorporates additional features to support the service offered by the AFTN. The Basic ATS Message

Service is further described in section 6.2.2.1.3. This includes the specification of which ISPs apply in this context.

6.1.2.1.3 The Extended ATS Message Service will provide functionalities in addition to those of the Basic ATS Message Service which are either one or several of the following:

a) functionalities which are optional in the ISPs applying in the context of the Basic ATS Message Service;

b) functionalities included in ISPs which do not apply in the context of the Basic ATS Message Service;

c) functionalities included in future editions of the ISO/IEC (International Organization for Standardization / International Electrotechnical Commission) and ITU-T (International Telecommunication Union - Telecommunications Standards) MHS Standards and Recommendations; and

d) functionalities included in future editions of the MHS ISPs.

6.1.2.1.4 An example of a) could be that the Extended ATS Message Service mandates the use of a Functional Group (e.g. Use of Directory) which is optional in the Basic ATS Message Service. An example of b) could be that the Extended ATS Message Service is based on a different category of service (e.g. Electronic Data Interchange Messaging (EDIMG)) defined in the MHS profiles. An example of c) could be the "Business Class Messaging extensions" currently under discussion at ISO and ITU-T, which define standard extensions to the Inter-Personal Message (IPM) Heading fields, among which some could potentially be used for the conveyance of items such as the filing time and the optional heading information currently carried in the ATS-Message-Header (i.e. in the body) of an AMHS IPM. The detailed specification of the Extended ATS Message Service is not included in these SARPs. It is for further study and inclusion in future issues of the SARPs.

6.1.2.1.5 The term ATS Message Service refers to the service which includes both the Basic and the Extended ATS Message Service where no distinction between these is necessary.

6.1.2.1.6 The ATS Message Service is the long-term solution amongst the ATS Message Handling Services defined over the ATN. This means that in the long-term, the ATS Message Service is aimed at becoming the single generic messaging service over the ATN.

6.1.2.2 *ATN Pass-Through Service Overview*

6.1.2.2.1 The ATN Pass-Through Service encapsulates and decapsulates AFTN messages at an AFTN/ATN type A Gateway.

6.1.2.2.2 Messages with multiple addressees are address-stripped in the AFTN/ATN Type A Gateway, and directed to different gateways as appropriate.

6.1.2.2.3 After determination of the destination gateway(s) as described above, the AFTN message is transparently conveyed without specific processing over the ATN Internet.

6.1.2.2.4 The upper layer protocol architecture used between two AFTN/ATN Gateways is the ATN Upper Layer Architecture as defined in Sub-Volume 4 of the SARPs.

6.1.3 References

References to other sections of Part III of this Comprehensive ATN Manual are given directly. References to sections of the Detailed Technical ATN SARPs [1] are noted as "section <n> of the SARPs". Other documents referenced in the text are indicated [n], where "n" is the number of the reference in the list below. Finally general references to multi-part standards are usually given in plain text.

[1] Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3 and Sub-Volume III of the *Manual of Technical Provisions for the ATN* (Doc 9705)

[2] Annex 10 — *Aeronautical Telecommunications*, Volume II

[3] CCITT Rec X.121 (1992) International numbering plan for public data networks

[4] CCITT Rec X.400 (1992). Message handling system and service overview

[5] CCITT Rec X.402 (1992). Message handling systems: overall architecture

[6] CCITT Rec X.411 (1992). Message handling systems: Message transfer system: Abstract service definition and procedures

[7] CCITT Rec X.419 (1992). Message handling systems: Protocol specifications

[8] CCITT Rec X.420 (1992). Message handling systems: Interpersonal messaging system

[9] ISO/IEC 646:1991. Information technology — ISO 7-bit coded character set for information interchange

[10] ISO/IEC 3166:1993. Codes for the representation of names and countries

[11] ISO/IEC 8859-1: 1987. Information processing — 8-bit single-byte coded graphics character sets — Part 1: Latin alphabet No. 1

[12] ISO/IEC TR 10000-1: 1995. Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework

[13] ISO/IEC TR 10000-2: 1995. Information technology — Framework and taxonomy of International Standardized Profiles — Part 2: Principles and taxonomy for OSI profiles

[14] ISO/IEC 10021-1:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 1 : System and Service Overview

[15] ISO/IEC 10021-1/Amd.2:1994. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 1 : System and Service Overview

[16] ISO/IEC 10021-2:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 2: Overall Architecture

[17] ISO/IEC 10021-2/Amd.1:1993. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 2: Overall Architecture

[18] ISO/IEC 10021-2/Amd.2:1994. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 2: Overall Architecture

[19] ISO/IEC 10021-3:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 3: Abstract Service Definition Conventions

[20] ISO/IEC 10021-4:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 4: Message Transfer System: Abstract Service Definition and Procedures

[21] ISO/IEC 10021-4/Amd.1:1994. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 4: Message Transfer System: Abstract Service Definition and Procedures

[22] ISO/IEC 10021-5:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 5: Message Store: Abstract Service Definition

[23] ISO/IEC 10021-5/Amd. 1:199x. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 5: Message Store: Abstract Service Definition

[24] ISO/IEC 10021-6:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 6: Protocol Specifications

[25] ISO/IEC 10021-7:1990. Information Technology — Text Communication — Message-Oriented Text Interchange System (MOTIS) — Part 7: Interpersonal Messaging System

[26] ISO/IEC ISP 10611-1:1994. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Common Messaging — Part 1: MHS Service Support

[27] ISO/IEC ISP 10611-2:1994. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Common Messaging — Part 2: Specification of ROSE, RTSE, ACSE, Presentation and Session Protocols for use by MHS

[28] ISO/IEC ISP 10611-3:1994. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Common Messaging — Part 3: AMH11-Message Transfer (P1)

[29] ISO/IEC ISP 10611-4:1994. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Common Messaging — Part 4: AMH12-MTS Access (P3)

[30] ISO/IEC ISP 10611-5:1994. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Common Messaging — Part 5: AMH13-MS Access (P7)

[31] ISO/IEC ISP 11188-1:1995. . Information Technology — International Standardized Profile — Common upper layer requirements — Part 1: Basic connection oreinted requirements

[32] ISO/IEC ISP 12062-1:1995. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Interpersonal Messaging — Part 1: IPM MHS Service Support

[33] ISO/IEC ISP 12062-2:1995. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Interpersonal Messaging — Part 2: AMH21 — IPM Content

[34] ISO/IEC ISP 12062-3:1995. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Interpersonal Messaging — Part 3: AMH22 — IPM Requirements for Message Transfer (P1)

[35] ISO/IEC ISP 12062-4:1995. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Interpersonal Messaging — Part 4: AMH23 — IPM Requirements for MTS Access (P3)

[36] ISO/IEC ISP 12062-5:1995. Information Technology — International Standardized Profiles AMH1n — Message Handling Systems — Interpersonal Messaging — Part 5: AMH24 — IPM Requirements for MS Access (P7)

6.2 ATS Message Service

6.2.1 System level

6.2.1.1 *ATS Message service users*

6.2.1.1.1 Two categories of users of the ATS Message Service are defined in the SARPs:

a) direct users; and

b) indirect users.

6.2.1.1.2 Direct users are those who make use of an ATS Message User Agent to access the ATS Message Service. The use of a User Agent (UA) gives them a potential access to:

a) the MHS Elements of Service supported in the Basic ATS Message Service (see 6.2.2.1.3); and

b) optional MHS Elements of Service in addition to those which are mandatory in the Basic ATS Message Service.

6.2.1.1.3 Direct users may belong to two subgroups as follows:

a) human users who interact with the ATS Message Service by means of a human-machine interface with an ATS Message User Agent connected to an ATS Message Server; and

b) host users which are computer applications running on ATN end systems and interacting with the ATS Message Service by means of application program interfaces (APIs). Such APIs are out of the scope of the SARPs.

6.2.1.1.4 Indirect users are those users located at an AFTN station which can only reach the AMHS via an AFTN/AMHS Gateway. Such users therefore have access only to the AMHS functionalities which have a direct equivalent in the AFTN.

6.2.1.2 *AMHS Model*

6.2.1.2.1 AMHS Functional model

6.2.1.2.1.1 Model components

6.2.1.2.1.1.1 The set of ATS Message Servers, ATS Message User Agents and AFTN/AMHS gateways is known collectively as the ATS Message Handling System (AMHS). The set of protocols implemented between ATS Message Servers and/or AFTN/AMHS Gateways is called the ATS Message Protocol Stack Type B. From the ATN Internet perspective, these three categories of systems are ATN End Systems.

6.2.1.2.1.1.2 Since the AMHS operates in a store-and-forward mode, the number of ATN End Systems involved in an end-to-end message transfer in the AMHS depends on each message being transferred, i.e. on its originator and recipient, as well as on the routing adopted for that message by the involved ATS Message Servers, at the moment of its conveyance.

6.2.1.2.1.1.3 In the case of a single message end-to-end conveyance in the AMHS, a number of ATS Message Servers and two systems among ATS Message UAs and AFTN/AMHS gateways are involved.

6.2.1.2.1.1.4 Following the concepts of the MHS Standards, it is necessary here to distinguish between the following components, or "building bricks", of the ATS Message Servers, and of the AFTN/AMHS gateways which all handle the ATS Message Protocol Stack Type B:

a) message transfer agent (MTA) which handles the "P1 protocol" (MTS transfer protocol) for the message exchange between a pair of MTAs. A set of interconnected MTAs forms a " message transfer system" (MTS);

b) user agent (UA) which is the interface between the user of the AMHS and the MTS. For the support of the Basic ATS Message Service, UAs provide the "interpersonal messaging (IPM) service", exchanging messages across the MTS from UA to UA by means of the "P2 protocol" (interpersonal messaging protocol);

c) message store (MS) which provides the MTA with a storage capability and which offers services allowing the UA to retrieve messages stored in the MS at its convenience. There are usually several UAs or MSs served by one MTA; and

d) access unit (AU) which in the AFTN/AMHS gateway provides the conversion capability supporting the interworking between the AFTN and the UAs of the AMHS. In the general MHS environment, AUs define how UA users can communicate with users of non-MHS technologies (e.g. telex). However, for the AMHS no use of such standardized AU types is made.

6.2.1.2.1.2 ATS Message Server overview

6.2.1.2.1.2.1 An ATS Message Server comprises a MTA and optionally one or several MSs. As far as upper layer MTA-to-MTA communications are concerned, i.e. above the transport layer, the SARPs only require compliance with the AMH22 Profile as specified in [34] and support of the IPM Distribution List Functional Group. This means that at this level, there is no " ATN-specific" requirement in the ATS Message Server specification. The interface between the ATS Message Server and the UAs it serves, either directly or through an MS, has been left open in the SARPs since this is often an implementation matter local to each Management Domain (see section 6.2.1.2.1.3 for more details).

6.2.1.2.1.2.2 If the ATS Message Server comprises any optional MS, then this MS is an IPM-MS. At the level of the IPM-MS the " ATN-specific" structured body is internal to the IPM body, and therefore it has no implication on the MS.

6.2.1.2.1.2.3 Figure 6.2-1 gives a simplified functional view of the ATS Message Server.

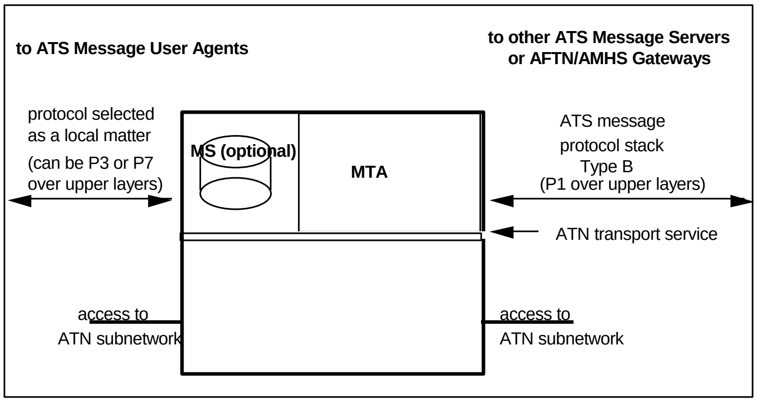

to ATS Message User Agents
to other ATS Message Servers or AFTN/AMHS Gateways
protocol selected as a local matter
(can be P3 or P7 over upper layers)
MS (optional)
MTA
ATS message protocol stack Type B
(P1 over upper layers)
ATN transport service
access to ATN subnetwork
access to ATN subnetwork

Figure 6.2-1. Functional view of the ATS Message Server

6.2.1.2.1.3 **ATS Message User Agent overview**

6.2.1.2.1.3.1 An ATS Message User Agent comprises a UA. In the Basic ATS Message Service, this UA is an IPM-UA which supports additional " ATN-specific" features in order to comply with the mandatory requirement of AFTN interworking. These additional requirements are related to the structure of the IPM body, they are detailed in 6.2.2.2.2.

6.2.1.2.1.3.2 As mentioned above, the interface between the ATS Message UA and the ATS Message Server at the level of upper layer UA-to-MTA communications (and vice-versa), i.e. above the transport layer, is a local implementation matter. The options at this level are as follows:

a) use the P3 protocol, if no MS is implemented in the ATS Message Server. In such a case, the use of the AMH23 Profile as specified in [35] is the preferred implementation choice;

b) use the P7 protocol, if MSs are implemented in the ATS Message Server. In such a case, the use of the AMH24 Profile as specified in [36] is the preferred implementation choice; and

c) use a locally-defined protocol, in the case of logically co-located UAs.

6.2.1.2.1.3.3 An ATS Message User Agent is by definition an ATN End System. The existence of this definition does not preclude the implementation, as a local matter, of UAs supporting a service identical to the Basic ATS Message Service without making use of the ATN for the interconnection between the UA and an ATS Message Server. Such UAs are also considered as logically co-located. In all cases logically co-located UAs in the AMHS are IPM-UAs supporting the structured IPM body defined in 6.2.2.2.2.

6.2.1.2.1.4 **AFTN/AMHS Gateway overview**

6.2.1.2.1.4.1 An AFTN/AMHS Gateway implements a MTA, and an AU. As further described in 6.2.3, the MTA forms the ATN Component of the AFTN/AMHS Gateway, and the AU is the Message Transfer and Control Unit of the AFTN/AMHS Gateway.

6.2.1.2.1.4.2 An AFTN/AMHS Gateway also implements an AFTN Component, however in strict terms this component does not pertain to the AMHS.

6.2.1.2.1.4.3 Finally an AFTN/AMHS Gateway includes a control position, which is the functional device where out-of-line situations and certain cases of non-delivery are handled, automatically and/or by a human operator. Since it is a local component dedicated to system management, the control position is only conceptually defined in the SARPs.

6.2.1.2.1.4.4 Figure 6.2-2 gives a functional view of the AFTN/AMHS Gateway.

6.2.1.2.1.5 **Interaction between AMHS systems**

6.2.1.2.1.5.1 Figure 6.2-3 illustrates different potential relationships together with examples of message flows between the systems which are part of the AMHS.

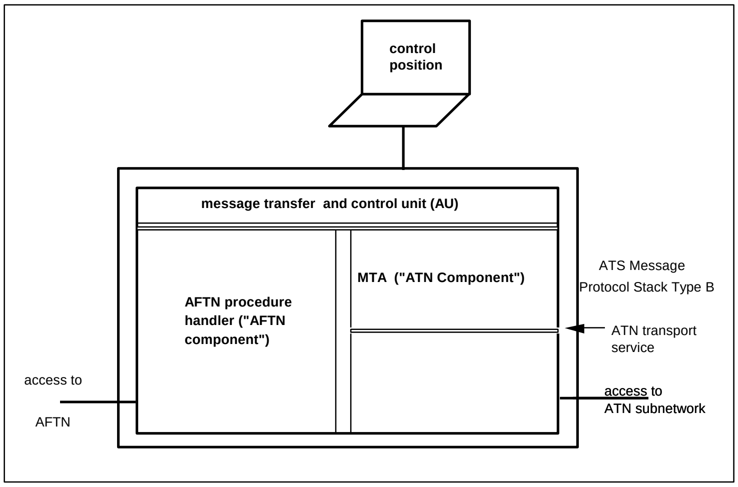

control position
message transfer and control unit (AU)
AFTN procedure handler ("AFTN component")
MTA ("ATN Component")
ATS Message Protocol Stack Type B
ATN transport service
access to
AFTN
access to ATN subnetwork

Figure 6.2-2. Functional view of the AFTN/AMHS Gateway

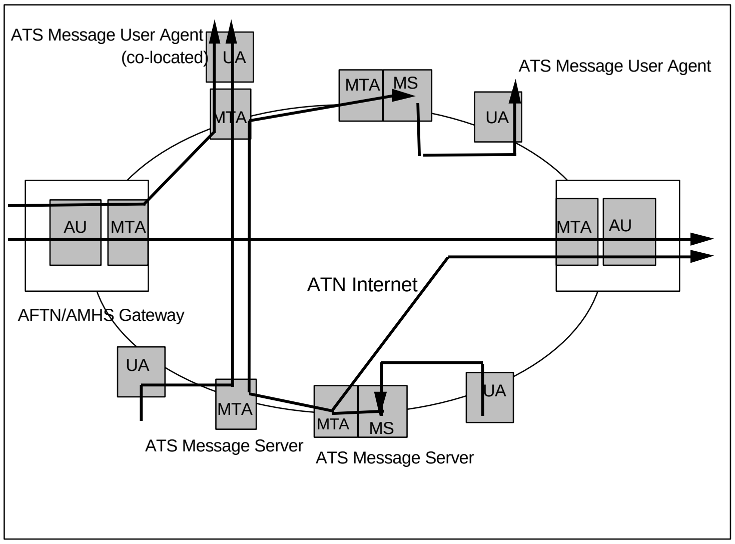


Figure 6.2-3. Examples of AMHS message flows

6.2.1.2.1.5.2 Other configurations are also possible, for example an MTA+AU+MS/UA may also be obtained by co-location of an AFTN/AMHS Gateway, an ATS Message Server and one or several UAs.

6.2.1.2.1.5.3 Figure 6.2-3 illustrates that functions such as routing ("relaying") and multiple dissemination of messages to different recipients are performed by the MTAs included either in an ATS Message Server or in an AFTN/AMHS Gateway.

6.2.1.2.2 AMHS information model

6.2.1.2.2.1 Information object types

6.2.1.2.2.1.1 In conformance with ISO/IEC 10021-2 three categories of information objects are conveyed in the AMHS: message, probe and report.

6.2.1.2.2.2 Messages

6.2.1.2.2.2.1 Messages are composed of two parts, the envelope and the content.

6.2.1.2.2.2.2 An envelope is generated by an ATS Message User Agent or an AFTN/AMHS Gateway when entering the AMHS. The envelope bears all the information necessary for the conveyance of the message by the ATS Message Servers towards its destination. The information carried by the envelope varies along the conveyance of the message towards its destination.

6.2.1.2.2.2.3 The type of envelope which is used for the submission/delivery of a message between an ATS Message User Agent and an ATS Message Server is related to the protocol implemented between the two systems. Therefore it is out of the scope of the SARPs (see 6.2.1.2.1.3). In consequence, the specifications included in the SARPs deal only with

Transfer Envelopes, used from MTA to MTA, i.e. either between two ATS Message Servers, between two AFTN/AMHS Gateways or between an ATS Message Server and an AFTN/AMHS Gateway.

6.2.1.2.2.2.4 The content of the message is an information object which the MTAs neither examine nor modify, except for conversion, during conveyance of the message. Messages generated in the Basic ATS Message Service are always Inter-Personal Messages (IPM). Two types of content conversion may be performed in the AMHS:

a) conversion of the content encoded-information-types, as specified in the base standards. Such a conversion is optional in the AMHS since there is no clause mandating the support of the Conversion Optional Functional Group as specified in the ISPs. It may be implemented in the MTAs, as a local matter; and

b) message content conversion in an AFTN/AMHS Gateway for a message conveyed from the AMHS to the AFTN. Such a conversion capability is necessary for interworking between the two messaging environments. It is further detailed in the AFTN/AMHS Gateway specification (see 6.2.3.5).

6.2.1.2.2.3 **Probes**

6.2.1.2.2.3.1 A probe is a class of message containing only an envelope which is conveyed by the MTAs from one user up to the MTA serving other users. It may be used to determine the deliverability of messages.

6.2.1.2.2.3.1.1 In the AMHS, probes are generated, if supported, at an ATS Message User Agent. An AFTN/AMHS Gateway does not generate probes. However, upon reception of probes, the AFTN/AMHS Gateway will process it and respond to it as appropriate.

6.2.1.2.2.4 **Reports**

6.2.1.2.2.4.1 A report is an information object generated by a MTA in order to report on the outcome or progress of a message or probe in the set of interconnected MTAs pertaining to the AMHS.

6.2.1.2.2.4.2 In the AMHS, reports are generated by an ATS Message Server or by an AFTN/AMHS Gateway. Within an AFTN/AMHS Gateway, the report may be generated either by the MTA comprised in the ATN Component (as usual for any MTA), or by the Access Unit (see 6.2.3.2.6).

6.2.1.2.3 **AMHS Security model**

6.2.1.2.3.1 The MHS standards include Elements of Service (EoS) related to security. However, for the support of the Basic ATS Message Service, their implementation is optional. It is expected that these EoS will be used in future versions of the SARPs to ensure the AMHS security.

6.2.1.2.3.2 Therefore, in the Basic ATS Message Service, security is deemed a local issue, to be addressed as appropriate at each ATN End System pertaining to the AMHS by the authority in charge of the system. It may be noted that also in the MHS standards and ISPs, certain Security EoSs such as the Access Management between UA and MTA are specified as “local matter”.

6.2.1.2.4 AMHS Management model

6.2.1.2.4.1 In the Basic ATS Message Service, management is limited to the logging provisions which are defined with two main goals:

a) ensuring message traceability, i.e. with the objective of keep track of the information objects which passed in, through and out of an ATN End System pertaining to the AMHS, and of the action taken thereon;

b) maintaining a long-term traffic log of the entire traffic upon origination, for safety and administrative purposes, e.g. in case an investigation would be necessary.

6.2.1.2.4.2 In the AFTN, this function is called long-term retention, and the retention duration is specified as being 30 days. Similar provisions are included in the ATSMHS SARPs, with the goal of offering the same level of functionality (traceability and originated traffic recording) as currently provided in the AFTN.

6.2.1.2.4.3 Within a given AMHS Management Domain (see 6.2.1.3.1 for the definition of this concept), the place where the originated traffic is recorded is a local matter. This may be done e.g. at the originating ATS Message Server (at its Submission/Delivery port), at the originating ATS Message User Agent, at the ATS Message Server where the message exits the AMHS Management Domain (at its exit Transfer port), or at a specifically dedicated system by ad-hoc means. At an AFTN/AMHS Gateway, there is no need to record the entire generated AMHS messages, since a message generated at an AFTN/AMHS Gateway as the result of the conversion of an AFTN message has already been logged in the AFTN for long-term retention.

6.2.1.2.4.4 For any piece of information, for which a logging requirement is present in the SARPs, the way in which the information specified is logged is an implementation matter, which is out of the scope of the SARPs. Also the way in which the information specified is retrieved, exchanged and used is an implementation matter, which is out of the scope of the SARPs.

6.2.1.3 *AMHS Organization*

6.2.1.3.1 AMHS Management Domains

6.2.1.3.1.1 For purposes of organization, addressing, routing etc. it is necessary to define an organizational structure for the AMHS.

6.2.1.3.1.2 MHS Standards require the organization of an MHS into domains which govern its management.

6.2.1.3.1.3 The organizational structure of the AMHS is aligned on these concepts without further refinement. This means that organizationally, the AMHS is made of AMHS Management Domains each of them compliant with the definition of a MHS Management Domain as may be found in the MHS standards.

6.2.1.3.1.4 Flexibility is given to the organizations participating in the AMHS, by the possible choice for an AMHS Management Domain to operate either as an Administrative Management Domain (ADMD) or as a Private Management Domain (PRMD).

6.2.1.3.1.5 Each AMHS Management Domain is responsible, among other things, for:

a) carrying out the relevant administrative procedures such as MD registration;

b) managing the equipment required to provide the ATS Message Service in its area of responsibility, among which at least one MTA, included either in an ATS Message Server or in an AFTN/AMHS Gateway;

c) managing the O/R (Originator/Recipient) Names and O/R Addresses (MF-Addresses (MHS-form Address) of all its service-users, allowing these users to be uniquely identified in the AMHS;

d) managing the routing internal to the Management Domain and the multilateral agreements related to inter-Management Domain routing;

e) performing the long-term logging of the entirety of messages (envelope and content) originated by its direct AMHS users; and

f) defining the various policies specified as a matter of local policy in the SARPs.

6.2.1.3.1.6 **Relations between AMHS Management Domains**

6.2.1.3.1.7 Each AMHS Management Domain must be interconnected over the ATN with at least one other AMHS Management Domain, which is then called “adjacent”. The concept of adjacent domains is not related to geographical considerations, but to a direct telecommunications relationship over the ATN between resources belonging to these organizations.

6.2.1.3.1.8 The communication between two AMHS Management Domains is always MTA to MTA, i.e. either:

a) from ATS Message Server to ATS Message Server;

b) from ATS Message Server to AFTN/AMHS Gateway; or

c) from AFTN/AMHS Gateway to AFTN/AMHS Gateway.

6.2.1.3.1.9 This means that the protocol implemented between two AMHS Management Domains is P1. Messages generated in the Basic ATS Message Service are IPMs including the structured body defined for the AMHS. However at the level of Message Transfer the structured body is not considered by the AMHS systems (ATS Message Server or AFTN/AMHS Gateway) involved in the "point-to-point" communication between the two AMHS Management Domains.

6.2.1.4 *AMHS Naming and Addressing*

6.2.1.4.1 AMHS Naming

6.2.1.4.1.1 Scope of AMHS naming

AMHS naming encompasses two different aspects:

a) naming of AMHS users, which is made by means of O/R names;

b) naming of the application processes and application entities in the ATN End Systems participating in the AMHS;

6.2.1.4.1.2 Naming of AMHS users

6.2.1.4.1.2.1 An O/R name identifies uniquely in the global MHS the name of a particular user. This name may take three forms, which are either the form of a Directory Name, or the form of an O/R Address, or both.

6.2.1.4.1.2.2 In the AMHS as defined in these SARPs, since the Use of Directory is optional in the Basic ATS Message Service, O/R names of AMHS users, when crossing the boundary between two AMHS Management Domains, can only take one of the forms comprising an O/R Address, which is denominated a MF-address in the AMHS (see 6.2.1.4.2.2).

6.2.1.4.1.3 Upper Layer naming

6.2.1.4.1.3.1 Each application entity participating in the AMHS may be identified with a unique name which is an Application Entity Title (AET). An Application Entity Title comprises an Application Process Title and an Application Entity Qualifier.

6.2.1.4.1.3.2 This AET may be used at the establishment of the association between two communicating MHS applications. It is an optional parameter of the A-Associate service primitive of the Association Control Service Element (ACSE), for both the calling entity and the called entity.

6.2.1.4.1.3.3 The value of the AET may be selected as a local matter, but recommended values for each category of AMHS end system are given in the SARPs. These values are defined consistently with the naming provisions for other applications as specified in section 4.3 of the SARPs.

6.2.1.4.2 **AMHS Addressing**

6.2.1.4.2.1 **Scope of AMHS Addressing**

Like naming, AMHS addressing encompasses two different aspects:

a) addressing of AMHS users, which is used for message routing from an AMHS user and delivery to another AMHS user, among the MTAs pertaining to the AMHS. This is made by means of O/R addresses; and

b) addressing of the upper layer entities in the ATN End Systems participating in the AMHS.

6.2.1.4.2.2 **Addressing of AMHS users**

6.2.1.4.2.2.1 Two address forms are defined to identify users in the AMHS, which are as follows:

a) an AF-Address (AFTN-form address) is used to locate AMHS users, either direct or indirect, in the AFTN address space; and

b) a MF-Address (MHS-form address) is used to locate a direct or indirect AMHS user in the AMHS address space.

6.2.1.4.2.2.2 An AF-Address is an ICAO AFTN 8-letter addressee indicator, as defined in Annex 10, Volume II.

6.2.1.4.2.2.3 A MF-Address is a MHS O/R address without particular restrictions or specifications other than those relative to the AMHS Management Domain which the user belongs to.

6.2.1.4.2.2.4 By definition, an indirect user has an AF-Address. If a direct user needs to communicate with indirect users, it is required that an AF-Address be allocated to him. The way in which this AFTN address is allocated is an administrative matter outside the scope of the SARPs.

6.2.1.4.2.2.5 The selection of the AMHS Addressing Scheme is usually a matter of policy local to each AMHS Management Domain. This addressing scheme may be either a local one or a Common AMHS Addressing Scheme, or a combination of these. Common AMHS Addressing Schemes are common schemes established at the level of ICAO. The adoption of a scheme, or the decision that every AMHS Management Domain within ICAO should use one or another Common AMHS Addressing Scheme is an institutional matter, which is therefore out of the scope of SARPs.

6.2.1.4.2.2.6 One single Common AMHS Addressing Scheme is defined in this version of the SARPs. It is called the XF-Addressing Scheme (translated-form) and it is the addressing scheme for indirect users, unless, for any particular reason, a more user-friendly O/R address (i.e. a MF-Address) is desired.

6.2.1.4.2.2.7 An XF-Address comprises exclusively the following attributes:

a) C (country-name) = either of the following:

1) two-character alphabetical country-indicator as specified in ISO 3166 [10];

2) three-digits data-country-code as specified in CCITT recommendation X.121 [3]; or

3) the two-letter alphabetical value reserved for international registration;

b) A (administrative-management-domain-name) = admd-name or single-space

c) P (private-management-domain-name) = prmd-name (present only if the AMHS Management Domain operates as a PRMD)

d) O (organization-name) = "AFTN"

e) OU1 (organizational-unit-names) = 8-letter addressee indicator (AF-address of the user).

6.2.1.4.2.3 **Upper Layer addressing**

6.2.1.4.2.3.1 Upper layer addresses include:

a) the Transport Service Access Point (TSAP) address which identifies the Transport Service-user, i.e. the session entity in an AMHS system. It comprises the Network Service Access Point (NSAP) address of the ATN End System complemented with a transport (T) Selector;

b) the Session Service Access Point (SSAP) address which identifies the Session Service-user, i.e. the presentation entity in an AMHS system. It comprises the TSAP address complemented with a session (S) Selector;

c) the Presentation Service Access Point (PSAP) address which identifies the Presentation Service-user, i.e. the presentation entity in an AMHS system. It comprises the SSAP address complemented with a presentation (P) Selector.

6.2.1.4.2.3.2 The allocation of the NSAP address obeys the rules defined in Sub-Volume 5. The allocation of T-, S- and P- selectors is considered a local matter for the organisation responsible for an AMHS system, and consequently for each AMHS Management Domain.

6.2.1.4.3 **Relationships between these concepts**

6.2.1.4.3.1 AMHS systems are by essence ground fixed systems. Therefore the mapping of an AET onto a PSAP address is unambiguous and static, unless in case of reconfiguration.

6.2.1.4.3.2 When trying to route an AMHS message (or probe, or report) in an ATS Message Server, the routing tables of the MTA are analysed to determine either of the following, based on the MF-Address of the message recipient:

a) the upper layer address of the recipient's UA, if the ATS Message Server is the delivering MTA, i.e. the last MTA in the sequence of MTAs in the end-to-end communication from UA (or Gateway) to UA (or Gateway); or

b) the upper-layer address of the next hop MTA if the recipient is not local to the current ATS Message Server.

6.2.1.4.3.3 In the first case, the UA's upper layer address which is found in the routing tables depends on the type of protocol implemented between the UA and the ATS Message Server, which is a local matter in the AMHS. In the latter case, the mapping, which is performed using the static MTA routing tables, usually derives a mta-name from the recipient O/R address, and the PSAP address corresponding to the mta-name. The AET of the next hop MTA, if configured, may also be found in the table. These parameters are used to establish an association, or use an existing one, with the determined next hop MTA.

6.2.1.4.3.4 When submitting an AMHS message (or probe) at an ATS Message User Agent, the situation is different since a UA usually communicates with one single MTA, which is always the same unless in case of reconfiguration. Therefore no mapping nor routing is required, since static parameters are simply configured and used in the considered UA.

6.2.2 **ATS Message Service Description**

6.2.2.1 ***Specification Scheme***

6.2.2.1.1 **Introduction to MHS Profiles**

6.2.2.1.1.1 The specifications on which the AMHS is based are very extensive and contain many functions which do not need to be implemented in the AMHS. For this reason, it is necessary to specify a profile which describes the functions to be included. Such profiles which have been standardized by ISO are known as ISPs (international standardized profiles).

6.2.2.1.1.2 Profiles standardize the use of options and other variations in the base standards, and provide a basis for the development of uniform, internationally recognized system tests. Implementations may then claim conformance with the ISPs, which in this way promote system interoperability without the users having to specify their own combination of functions among those made available by the base standards.

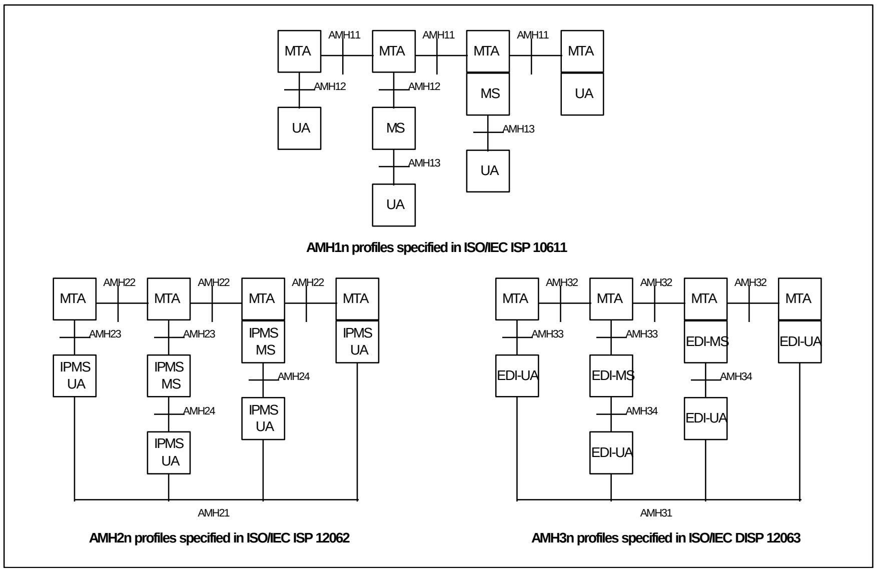
6.2.2.1.1.3 ISPs are classified in ISO/IEC Technical Report (TR) 10000, which is the Framework and Taxonomy of International Standardized Profiles. In this document, the ISO MOTIS (Message Oriented Text Interchange Systems, the initial ISO name for MHS) are arranged under Application Profiles: Message Handling (AMH). For Common Messaging, i.e. for the Message Transfer System (MTS), for the MTS-Access and for the MS-Access the profiles AMH1n (n=1 to 3) are relevant; for the Interpersonal Messaging Service (IPMS) the profiles AMH2n (n=1 to 4) are relevant, while for the Electronic Data Interchange Messaging Service (EDIMG Service) the profiles AMH3n (n=1 to 4) are relevant. Figure 6.2-4 depicts, for each of the AMH set of profiles, where each AMHnn applies.

6.2.2.1.1.4 Additionally, each of the ISPs includes a first part to describe the overall specifications of the support of the Elements of Service (EoS) and associated functionalities which are not appropriate for consideration only from the perspective of a single MHS protocol.

6.2.2.1.1.5 In the context of the Basic ATS Message Service, the AMH2n set of Profiles are those which are applicable.

6.2.2.1.2 **Classification of requirements**

6.2.2.1.2.1 The specification scheme is based on sets of Elements of Service (EoS). An EoS is a well-defined MHS function provided by a MHS functional object such as MTA, UA, MS or AU or by the MTS (i.e. the set of interconnected MTAs) and is defined in ISO/IEC 10021. An element of service usually leads to the inclusion of specific fields in the protocol data units.



AMH1n profiles specified in ISO/IEC ISP 10611

AMH2n profiles specified in ISO/IEC ISP 12062

AMH3n profiles specified in ISO/IEC DISP 12063

Figure 6.2-4. Applicability of AMH Profiles

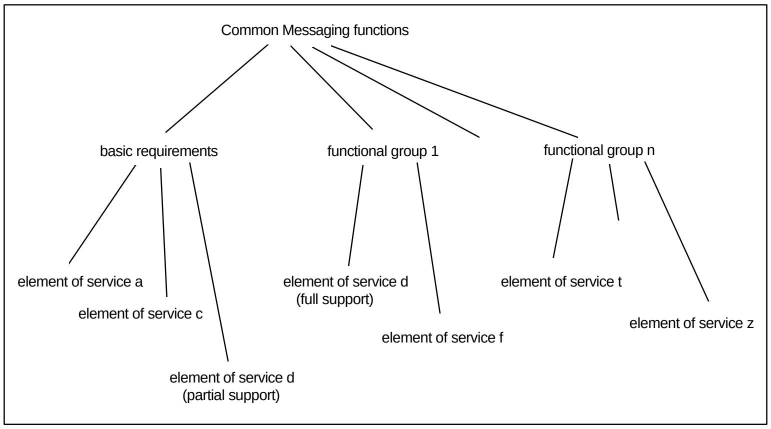


Figure 6.2-5. Relationship of elements of service and functional groups

6.2.2.1.2.2 The AMHS profiles make reference to Part 1 of the ISPs for the general specification of the supported EoS, and also to the relevant AMH profiles for the protocols supported by the AMHS.

6.2.2.1.2.3 The ISPs define the terms "basic requirements" and "functional group". The "basic requirements" are Elements of Service and associated features (e.g. protocol elements) which are required to be supported by all MHS implementations. A "functional group" is a set of one or several EoS which are related to each other, and the associated features, which together support a significant optional area of MHS functionality.

6.2.2.1.2.4 An EoS which is part of a functional group may be optionally supported by an implementation claiming only conformance to the basic requirements. On the other hand, an implementation claiming conformance for support of the optional functional group supports it as a whole, i.e. all EoS and associated features of the functional group are implemented.

6.2.2.1.2.5 In some cases, the partial support of an EoS may be included in the basic requirements, while its "full support" is part of an optional functional group. This may happen for example, to allow the proper end-to-end "transport" of a functionality across the MTS when this optional functionality is implemented.

6.2.2.1.2.6 The basic requirements together with the complete set of optional functional groups, as specified in ISO/IEC ISP 10611 (Common Messaging), make up the complete set of MHS functions related to common messaging, i.e. non content-dependent specific functionalities. This is illustrated in Figure 6.2-5.

6.2.2.1.2.7 The elements of service of the optional functional groups may be implemented, but by definition they do not have to be implemented. If they are implemented, then the implementation must conform to the base definitions in ISO/IEC 10021 and to the clauses of ISO/IEC ISP 10611 and the EoS must be treated as if it were specified as mandatory support. If they are not implemented, then the functions corresponding to the elements of service are simply not carried out. However the absence of the functions may not cause a protocol error to be generated when a protocol data unit referring to a non-implemented element of service is received. This requirement allows a basic compatibility among all ATS Message Servers even when these have different levels of functionality, for example between the ATS Message Servers which implement the Basic ATS Message Service and those which, in the future, will implement the Extended ATS Message Service. Such optional functional groups could, for example, be usefully employed within an area administered by one authority (AMHS Management Domain) or between pairs of AMHS Management Domains based on bilateral agreements.

6.2.2.1.2.8 It is expected that in the future, i.e. in the Extended ATS Message Service, the Security (SECn) and Use of Directory (DIR) Functional Groups could be used, since they would bring useful functionality to ensure the AMHS security and to ease the management of O/R names.

6.2.2.1.3 AMHS service characteristics for support of the Basic ATS Message Service

6.2.2.1.3.1 As already introduced in section 6.2.1.2.1, the AMHS includes a set of UAs, AUs together with the MTS. When supporting the Basic ATS Message Service, the service performed by UAs and AUs is the Interpersonal Messaging Service (IPM Service) as defined in the MHS Standards.

6.2.2.1.3.2 The AMH21 Profile, as specified in [33], applies on an end-to-end basis between the UAs and, by extension, the AUs belonging to the AMHS, which are implemented in the ATS Message User Agents and AFTN/AMHS Gateways, and which support the Basic ATS Message Service.

6.2.2.1.3.3 The IPM Service characteristics, as supported by the AMH21 Profile for the requirements of the AMHS in the context of the Basic ATS Message Service, are described in the context of the ATS Message User Agent in section 6.2.2.2. This description also includes the additional requirements necessary for interworking with the AFTN.

6.2.2.1.3.4 The AMH22 Profile, as specified in [34], applies between ATS Message Servers, between an ATS Message Server and an AFTN/AMHS Gateway, and between two AFTN/AMHS Gateways. It may be noted that this implies that the AMH11 Profile, as specified in [28], is also applicable between the ATS Message Servers and AFTN/AMHS Gateways.

6.2.2.1.3.5 The MT (Message Transfer) Service characteristics, as supported by the AMH22 Profile for the requirements of the AMHS in the context of the Basic ATS Message Service, are described in the context of the ATS Message Server in section 6.2.2.3.

6.2.2.1.3.6 The use of the AMH Profiles as presented above is illustrated in Figure 6.2-6.

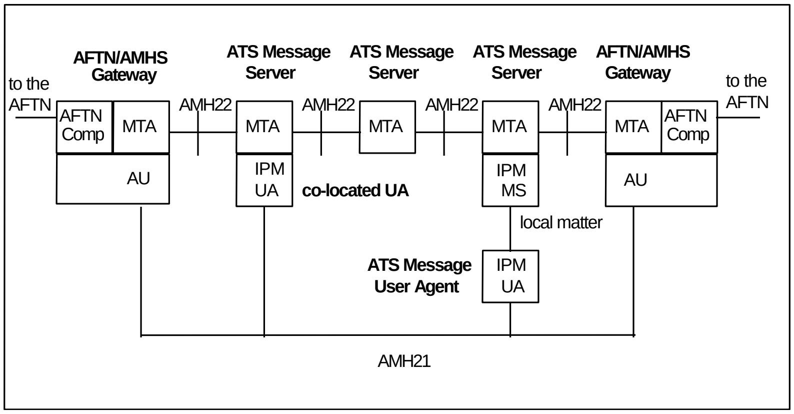


Figure 6.2-6. Use of AMH Profiles in the AMHS

6.2.2.2 *ATS Message User Agent Profile Description*

6.2.2.2.1 General characteristics

6.2.2.2.1.1 The AMHS Profile for an ATS Message User Agent includes only the specification of the Message Content, i.e. the support of AMH21 as introduced in 6.2.2.1.3, and additional requirements related to the interworking with the AFTN.

6.2.2.2.1.2 These additional requirements are related to:

a) the contents of the ia5-text or general-text body part; and

b) the support of receipt-notification-requests which is mandatory on origination, while it is optional in AMH21.

6.2.2.2.1.3 If conformance to CCITT X.400-84, or interworking with X.400-84 IPM UAs is required, this implies for the ATS Message User Agent the additional support of the IPM 84 Interworking (84IW) Functional Group. Although not mandated by the SARPs, such support is encouraged since it makes possible the interworking with IPM UAs not supporting content-type 22. It is recalled that while the base standards (or at least the versions of the base standards which are applicable to the ATN) include procedures for downgrading P1 protocol elements from CCITT X.400-1988 to CCITT X.400-1984, there are no such provisions for downgrading an IPM format with content-type 22 (inter-personal-messaging-1988) to an IPM format with content-type 2 (inter-personal-messaging-1984). Nevertheless Annex C to [32], which is informative, provides some additional recommendations for this purpose.

6.2.2.2.1.4 When an ATS Message User Agent generates an IPM for transfer in the AMHS, the use of ia5-text body part and of content-type 2 is encouraged if the text of the message contains only IA-5 characters. The reason is that interoperability with both 1984 and 1988 IPM UAs is then ensured, while this would not be true if an extended body-part type, either extended ia5-text or general-text, were used. The base standards themselves (see

Section 20.2 of [25], Note 2) favour content-type 2 whenever possible, to foster interworking with systems conforming only to CCITT Recommendation X.420 (1984).

6.2.2.2.2 **Body part contents**

6.2.2.2.2.1 As mentioned in 6.2.1.2.1.3, an ATS Message User Agent uses a structured body part to convey message components which are necessary for AFTN interworking.

6.2.2.2.2.2 This structured body part comprises:

a) an ATS-Message-Header element, which conveys the AFTN parameters which have no direct equivalent in MHS standards; and

b) an ATS-Message-Text element, which conveys the text of the message itself.

6.2.2.2.2.3 The parameters conveyed by means of the ATS-Message-Header are the following:

a) priority indicator, which is conveyed in a structure called ATS-Message-Priority;

b) filing time, which is conveyed in a structure called ATS-Message-Filing-Time; and

c) optional-heading-information, which is conveyed in a structure called ATS-Message-Optional-Heading-Info.

6.2.2.2.2.4 For conformance with the SARPs, an ATS Message User Agent must include the static capability to support these parameters. This means that the ATS Message User Agent must be able to generate the mandatory elements, which are the ATS-Message-Priority and the ATS-Message-Filing-Time, and may be able to optionally generate the ATS-Message-Optional-Heading-Info. Like for most of the IPM Heading Fields, the ATS Message User Agent is not mandated to generate these parameters for each submitted message, but only to have the capability to generate them. However, the ATS-Message-Priority and the ATS-Message-Filing-Time parameters are mandatory for messages directed to the AFTN, and their absence in a message will cause rejection at an AFTN/AMHS Gateway.

6.2.2.2.2.5 The ATS-Message-Header is composed of uppercase IA5IRV characters, including prompts to allow a reader to identify easily the included parameters. When displayed using a human-machine-interface which does not interpret the ATS-Message-Header, the external appearance of the body of an AMHS Message would be as in the following example:

PRI: FF
FT: 281120
OHI: DEFG2345... (if present)
(blank line)
(Beginning of message text)

6.2.2.2.2.6 Furthermore the ATS-Message-Header starts with a non-printable character which is SOH (which may be typed in, if required, using the Alt-1 keys in an MS-DOS or Windows environment) and ends with another non-printable character which is STX (which may be typed in, if required, using the Alt-2 keys in an MS-DOS or Windows environment).

6.2.2.2.2.7 This structured header may be generated by different means, such as:

a) it may be directly typed in within the body part, by a direct user at an off-the-shelf UA. This allows to use standard off-the-shelf UAs with their default human-machine interface without particular additions for the AMHS;

b) it may be generated by an additional input/display grid placed in the human-machine interface of the UA. In such a case the user would for example only type in the value of the priority-indicator and of the filing-time. Syntactic checks on these values may also be incorporated in the add-on in this case;

c) other approaches are possible, e.g. to generate automatically the filing time, etc.

6.2.2.2.2.8 The reasons for the conveyance of these parameters are the following:

a) there is a need for complete transparency for messages conveyed in the AFTN, then converted to the AMHS, and then converted back to the AFTN;

b) the AFTN priority indicator has five possible values, which bear different semantic meanings, and which are therefore not strictly equivalent to the three MHS priority levels, although in the AFTN there are only three transmission priority levels;

c) in the Aeronautical Fixed Service (AFS), the filing time bears a semantic value which may be different from that of the MHS submission-time;

d) for interworking purposes, there is a need to convey towards the AMHS message recipient the optional heading information, if present, which was carried in the AFTN Heading of a message converted from the AFTN to the AMHS.

6.2.2.2.3 Use of priority-indicators and notification-requests

6.2.2.2.3.1 In the AMHS, the MHS priority value “urgent” is reserved for distress messages, i.e. messages with the highest priority level, whose priority indicator is “SS” in the AFTN and the ATS-Message-Priority element.

6.2.2.2.3.2 Notification requests are used exclusively for messages with this highest priority level, in line with the principles adopted in the AFTN, where positive message acknowledgements only exist for SS priority messages. In such a case the notification-request parameter takes the value “rn”. For this purpose, an ATS Message User Agent must be able to generate such a notification request, although this is only optional in AMH21.

6.2.2.2.3.3 This means that three parameters are correlated in an AMHS message, and may be used only in conjunction with one another. They are the following:

a) the MHS priority element of the Message Transfer Envelope;

b) the priority-indicator in the ATS-Message-Header; and

c) the notification-requests in the primary-recipients, copy-recipients and blind-copy-recipients fields of the IPM Heading.

6.2.2.2.3.4 The mapping table between the MHS priority, which may take three different values, and the AFTN priority (or priority-indicator in the ATS-Message-Header), which may take five different values, is expressed in Table 6-2.1, where each row represents a valid set of parameters to be used together in a given AMHS message, depending on the message category as defined in Annex 10, Volume II.

Table 6.2-1. Message priorities and receipt notifications

message category	priority-indicator value (in the ATS-Message-Header)	MHS priority element value (in the Message Transfer Envelope)	notification-request value (in each recipient-specifier in the IPM Heading)
distress messages	SS	urgent	rn
urgency messages	DD	normal	default (none), nrn or ipm-return
flight safety messages	FF		default (none), nrn or ipm-return
meteorological messages, flight regularity messages, aeronautical information services (AIS) messages	GG	non-urgent	default (none), nrn or ipm-return
aeronautical administrative messages	KK	non-urgent	default (none), nrn or ipm-return

6.2.2.2.3.5 The correlation between these parameters may be done either automatically, using the add-on functionality implemented at the human-machine interface of a UA, or manually, with the potential of generating errors if the consistency is not properly ensured by the human end-user. When selecting the principle of an add-on functionality or of manual action, it should be recalled that a software add-on functionality needs to be maintained, particularly when the version of the underlying UA off-the-shelf software evolves.

6.2.2.2.3.6 If a message is generated, which does not meet the correlation requirements expressed above, the following happens in the AMHS:

a) there is no impact at ATS Message Servers;

b) if the message is directed to an ATS Message User Agent, there is no provision in the SARPs preventing its delivery by the last ATS Message Server nor its receipt by the user. Rejection by the ATS Message User Agent because of mismatch between these parameters should be avoided. If implemented in contradiction with the previous recommendation, such rejection is purely a local matter and may only happen at the reception stage (i.e. generation of a non-delivery report would contradict the base standards);

c) if the message reaches an AFTN/AMHS Gateway, and has a priority-indicator which is “SS”, the message is converted and forwarded, and additionally an error is reported to the control position in the following cases:

   1) if a receipt-notification is not requested (the abstract-value of the notification-request does not include “rn”); and

   2) if the MHS message priority differs from “urgent”;

d) if the message reaches an AFTN/AMHS Gateway, and has a priority-indicator which differs from “SS”, there is no impact of any mismatch between the parameters:

   1) the message is converted into an AFTN message;

   2) the priority used for conveyance in the AFTN is the priority-indicator, while the MHS priority is ignored;

   3) the receipt-notification-request, if any, is ignored.

6.2.2.3 *ATS Message Server Profile Description*

6.2.2.3.1 **Application profile**

6.2.2.3.1.1 The AMHS Profile for an ATS Message Server includes only the specification of the AMH profile as specified in ISO/IEC ISP 12062, which in turn implies several conformance requirements, in accordance with the principles described in 6.2.2.1.

6.2.2.3.1.2 The applicable profile is AMH22, which implies conformance with AMH111. The only additional requirement relates to the mandatory support of the IPM Distribution List (DL) Functional Group, so as to include in the AMHS a functionality equivalent to that of Pre-Determined-Addressee-Indicators (PDAIs) in the AFTN.

6.2.2.3.1.3 An important option, which is left as a matter of policy local to each AMHS Management Domain, is the question of the conformance to ITU-T X.400. An AMHS Management Domain may be required to conform to ITU-T X.400, e.g. under the following circumstances:

a) to comply with national regulation when registration by the national registration authority is requested; and

b) to interconnect with public MHS ADMDs which are by definition CCITT X.400 (84 or 88) and/or ITU-T X.400 (92 and later) compliant.

6.2.2.3.1.4 If conformance to ITU-T X.400 is required, this implies for the ATS Message Server the additional conformance to Profile AMH112. Support of AMH112 corresponds to the additional support of the mts-transfer-protocol and mts-transfer-protocol-84 application contexts, and to the support of the 84-Interworking (84IW) Functional Group.

6.2.2.3.1.5 A further consequence of the support of AMH112 is that the support of the Reliable Transfer Service Element (RTSE) and of the Association Control Service Element (ACSE) in X.410-84 is then required.

6.2.2.3.2 **Upper Layer Profile**

6.2.2.3.2.1 The support of AMH11 implies compliance with ISO/IEC ISP 10611-2[27]. This part of MHS ISPs specifies the Remote Operation Service Element (ROSE), RTSE, ACSE, Presentation and Session Protocols for use in the MHS.

6.2.2.3.2.2 For an ATS Message Server, [27] specifies the use of RTSE, which includes the support of monologue dialogue-mode, checkpointing, and as a minimum normal mode. As mentioned previously, X.410-1984 mode may also be required.

6.2.2.3.2.3 Concerning ACSE, presentation and session protocols, [27] also mandates conformance to ISO/IEC ISP 11188-1 [31] (also known as Common-Upper-Layer-Requirements-1 or CULR-1 profile), with some additional requirements specified in Annex A of [27].

6.2.2.3.2.4 An important addition to CULR-1 is in the session protocol, where several functional units are mandatory for the support of P1 (half duplex, minor synchronize, exceptions and activity management). This requirement is one of those which justified the non-use in the AMHS of the ATN Upper Layer Protocol Architecture defined in Sub-Volume 4 of the SARPs, since this architecture does not support these session functional units.

6.2.2.3.3 Use of the Transport Service

6.2.2.3.3.1 Use of the ATN Transport Service

6.2.2.3.3.1.1 An ATS Message Server by definition uses the ATN Transport Service to communicate with other ATS Message Servers.

6.2.2.3.3.1.2 Several parameters need to be given to the transport service provider, when requesting a transport connection to be established. These parameters are specified in Sub-Volume 5 of the SARPs. For most of these parameters, a single value is selected, either in the SARPs or as a local matter, to be used when establishing a transport connection between two ATS Message Servers.

6.2.2.3.3.1.3 More specifically, the base MHS standards used in these SARPs do not allow for the establishment of different transport connections with different quality of service parameters, based on the distinction between application level MHS priorities. This is due to the absence of a QoS parameter in the MTA-Bind abstract-operation and in the RT-OPEN service. Thus a single transport priority, conveying messages with different application-level priorities is used.

6.2.2.3.3.1.4 The way to request the use of the specified parameters to the Transport Service provider is an implementation matter which is out of the scope of the SARPs.

6.2.2.3.3.2 Use of the Transport Service for the AMH112 Profile

6.2.2.3.3.2.1 If profile AMH112 is supported, then the ATS Message Server shall implement an ISO 8073 Class 0 Transport protocol (TP0) over X.25. This cannot be implemented over the ATN, it is therefore out of the scope of the SARPs. However it is required, for example, if interconnection with a public X.400 ADMD is the local policy of a given AMHS Management Domain.

6.2.2.3.3.2.2 In such a situation, the co-existence of the support of Classes 0 and 4 of the ISO 8073 Transport protocol is an implementation matter, which is out of the scope of the SARPs.

6.2.2.3.3.2.3 Furthermore, the parameters specified in 6.2.2.3.3.1 concerning the use of the ATN Transport Service are not applicable in such a context.

6.2.2.3.3.3 **Implementation options**

6.2.2.3.3.3.1 For those MHS off-the-shelf implementations which do not intrinsically support the use of the ATN Transport Service, the AMHS compliant upper layers and application entities may be integrated as follows with the lower layers of an ATN end system.

6.2.2.3.3.3.2 At the lower boundary below the AMHS upper layers and application implementation, a transport service interface may be specified to intercept the transport service primitives, and to map these onto ATN Transport Service primitives using the intercepted data and additional parameter values which cannot be passed from the AMHS upper layers.

6.2.2.3.3.3.3 The consolidated ATN Transport Service primitive can then be passed to the transport service provider which provides the ATN Internet Communication Services.

6.2.2.3.3.3.4 The parameters required by the ATN Transport Service Provider for the establishment of an ATN transport connection are specified in section 5.5.1 of the SARPs. They are as follows:

a) called and calling TSAP addresses;

b) whether or not the expedited data option is required;

c) the required residual error rate (RER) to determine whether or not non-use of the transport checksum is allowed;

d) the Transport Connection Priority to be mapped into the resulting CLNP (Connectionless Network Protocol) NPDUs (Network Protocol Data Units); and

e) the ATN Security Label.

6.2.2.3.3.3.5 The ATN Security Label and the requested Transport Connection Priority are examples of additional parameters which cannot be passed from the AMHS upper layers.

6.2.2.3.3.3.6 Such an implementation architecture is depicted in Figure 6.2-7.

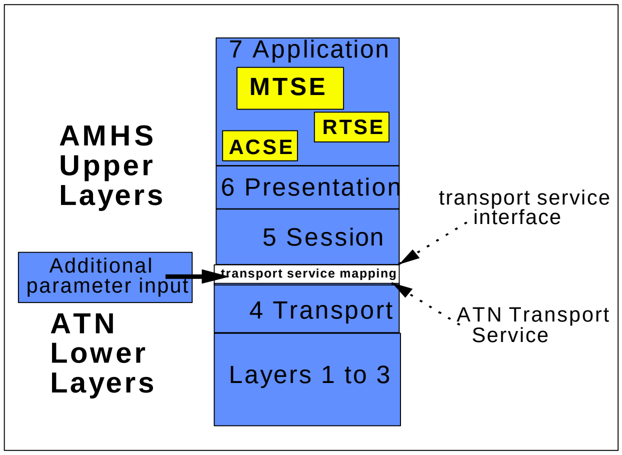


Figure 6.2-7. AMHS over ATN Transport Service implementation architecture

6.2.2.3.3.3.7 A similar mapping mechanism between a lower boundary interface and the transport service interface offered by the actual transport service providers may also be implemented in case of co-existence of different transport protocol stacks within a single system (e.g. ATN Transport Service and ISO TP0 over X.25 for connectivity towards a public ADMD).

6.2.2.3.4 Logging functions at an ATS Message Server

6.2.2.3.4.1 The SARPs specify the minimum logging requirements at an ATS Message Server. These long-term logging requirements are related to the administrative and legal requirement for the record of communications as specified in Annex 10, Volume II, 3.5. The SARPs requirements make it possible to perform message tracking through the AMHS, for example when an investigation is needed.

6.2.2.3.4.2 However, for an efficient management of the AMHS systems within one AMHS Management Domain, it could be useful to record more information, in particular about events not directly related to message transmissions but in relation with a good system operations.

6.2.2.3.4.3 The following events may be recorded for such management purposes:

a) MTA-bind (to or from another MTA) operation successful completion;

b) MTA-unbind (to or from another MTA); and

c) MTA-bind (to or from another MTA) error.

6.2.2.3.4.4 The information recorded in relation with the events above may include the following parameters which are either arguments, results or errors of the abstract operations:

a) initiator-name (if present);

b) initiator-credentials (if present);

c) security-context (if present);

d) responder-name (if present);

e) responder-credentials (if present); and

f) bind-errors (if any).

6.2.2.3.4.5 Additionally, to maintain a traffic log reflecting the entire traffic flows through an ATS Message Server, and actions taken by an ATS Message Server, it may be required to record events related to the following events:

a) Probe Submission operation (successful or error);

b) Probe Transfer in successful operation;

c) Probe Transfer out operation (successful or error);

d) Message Submission error;

e) Message Delivery error;

f) Report Delivery error.

6.2.3 **AFTN/AMHS Gateway Description**

6.2.3.1 ***General presentation***

6.2.3.1.1 A gateway is a communication device that permits message traffic to be transferred between two dissimilar communication systems. The gateway must perform its special operations transparently to function as an ordinary device in each of the two interconnected communication systems.

Note.— In the following, the AFTN/AMHS Gateway is modelled as a stand-alone facility. The alternative integration in an AFTN Centre is considered as an implementation matter.

6.2.3.1.2 The AFTN/AMHS Gateway has been conceived and designed as a technical tool to facilitate the transition from the world-wide AFTN to the new technology of ATS Message Handling Services over the ATN, namely the AMHS. To be useful, the gateway must be simple and reliable. It must also be easy to develop, deploy and maintain until it has achieved its purpose and has been withdrawn from use. Every effort has been directed towards developing a conceptual design that is fundamentally complete and does not attempt to correct AFTN deficiencies or to provide for services that are not absolutely required.

6.2.3.1.3 The AFTN/AMHS gateway is a device that must live in two worlds at once; the AFTN and the AMHS. In order to achieve this it must have two essential characteristics: transparency and isolation. Transparency allows the gateway to hide the fact that it is a gateway altogether and to appear to be a normal member of the AFTN and of the AMHS. Isolation hides the existence of each messaging system from the other.

6.2.3.2 *Functional breakdown*

6.2.3.2.1 **Functional model**

6.2.3.2.1.1 The functional model used to define the requirements for the AFTN/AMHS gateway has been presented in Figure 6.2-2. This model provides an abstract view that facilitates the definition of the components of the gateway and the assignment of functions to each component.

6.2.3.2.1.2 The three major components of the gateway, the AFTN Component, the ATN Component and the message transfer and control unit, are interconnected as shown in Figure 6.2-2 to provide an architecture that assures isolation and transparency. The functions assigned to each component are presented in the following sections.

6.2.3.2.2 **AFTN Component**

6.2.3.2.2.1 The access to AFTN in Figure 6.2-2 represents a point of connection to an external AFTN Centre. Send and receive functions are incorporated into the AFTN Component to establish a complete AFTN circuit connection to the AFTN Centre. The gateway must provide a sufficiently complete set of AFTN procedures to appear to be an AFTN Station. This imposes several special requirements and restrictions. Some functions in addition to those of an AFTN Station are necessary in the AFTN Component, due to its particular status as part of an AFTN/AMHS Gateway. However this does not alter the external appearance of the AFTN Component, as it may be seen from the AFTN Centre to which it is connected.

6.2.3.2.2.2 An AFTN address must be allocated to the AFTN Component, it is required in particular for the handling of the AFTN procedure between the AFTN Component and the AFTN Centre to which it is connected. Also to appear as an AFTN Station, this address is equally required.

6.2.3.2.2.3 Since the AFTN Component operates in a manner which is indistinguishable from an AFTN Station by the AFTN Centre to which it is connected, the AFTN Component is not required to have any diversion routing capability. Diversion routing is generally implemented in the AFTN Centre to which the AFTN/AMHS Gateway is connected. An implication of this situation is that an AFTN/AMHS Gateway is connected to a single AFTN Centre, if communicating with the AFTN side through an AFTN circuit.

6.2.3.2.2.4 However, from an implementation viewpoint, it is likely that in many occasions an AFTN/AMHS Gateway will be co-located with an AFTN Centre. Such a co-location may also be logical, which means that the AFTN/AMHS Gateway and the AFTN Centre do not communicate through an AFTN circuit, but rather using ad-hoc procedures eg. on a local area network. In such a case, some of the functions specified for the AFTN Component may not be required (eg. discarding of channel-check transmissions). It is then sufficient that:

a) the co-located AFTN Component and the AFTN Centre together fulfill the required functions;

b) the AFTN Component provides the Message Transfer and Control Unit of the AFTN/AMHS Gateway with an interface identical to that specified in the SARPs.

6.2.3.2.3 **ATN Component**

6.2.3.2.3.1 The ATN Component allows the gateway to function as an end system on the ATN. It incorporates an MTA in a manner equivalent to that of an ATS Message Server.

6.2.3.2.3.2 This MTA must implement the DL Functional Group, in compliance with the ATS Message Server specification. If the AMHS Management Domain operating the AFTN/AMHS Gateway additionally desires to implement other optional Functional Groups, this may be done in the ATN Component. For example, the ATN Component is the part of an AFTN/AMHS Gateway where the AMHS rerouting and/or redirection capability of the gateway, if any, is implemented.

6.2.3.2.4 **Message Transfer and Control Unit**

6.2.3.2.4.1 The remaining component, as shown in Figure 6.2-2, is named the Message Transfer and Control Unit (MTCU). In an AFTN/AMHS Gateway, this is the MHS Access Unit (AU) which provides application level functions that are not part of either the AFTN Component or of the ATN Component. These functions bind and integrate the other two components and are essential to the operation of the gateway. They include:

a) general provisions, which themselves cover two main subjects:

1) traffic logging,

2) address look-up tables which include the information necessary for the address conversion process between the two address spaces of the AMHS and of the AFTN to be performed;

b) AMHS to AFTN conversion, for the conversion of information objects received from the AMHS for potential conveyance in the AFTN. Because the AMHS level of functionality is higher than that of the AFTN, this function includes all the necessary processing to determine the gateway ability to convert the information object, and the necessary actions related to the potential rejection if the AFTN cannot convey the received information object;

c) AFTN to AMHS conversion, for the conversion of messages received from the AFTN for potential conveyance in the AMHS. For isolation purposes,the AFTN/AMHS Gateway converts in an automated manner only those AFTN service messages which have an end-to-end significance and which have an equivalent in the AMHS.

6.2.3.2.4.2 Although it communicates with the ATN Component through a transfer interface (see 6.2.3.2.6), the Message Transfer and Control Unit is not required to implement the functionalities associated with any of the optional functional groups defined in the ISP. More specifically the MTCU is not supposed to perform any DL-expansion. If supported in an AFTN/AMHS Gateway, such functionalities are implemented in the ATN Component, which incorporates a MTA equivalent to that of an ATS Message Server.

6.2.3.2.5 **Control Position**

6.2.3.2.5.1 Each gateway also includes an operator control position, or other input-output devices to accomplish the same function. The control position provides a method to load, initialize and control the operation of the gateway. The terminal is also used to display or record transient conditions and out-of-line situations, including error reports related to the gateway processing. Finally it is also the place where AMHS non-delivery reports are conveyed, when they cannot be processed in an automated manner by the gateway.

6.2.3.2.5.2 The control position enables interventions by the operator, permitting bidirectional communication with the human operator.

6.2.3.2.5.3 In summary, the control position therefore provides an operator interface where exception cases which could not be handled in an automated manner by the AFTN/AMHS Gateway components, may be handled and reacted upon. Also, it is a matter of policy local to the AMHS Management Domain operating the AFTN/AMHS Gateway, to decide whether certain categories of exception cases are handled automatically or with the assistance of the control position.

6.2.3.2.5.4 The format used by the AFTN Component, the ATN Component and the Message Transfer and Control Unit of an AFTN/AMHS Gateway to report errors and to convey non-delivery reports to the control position is a matter of policy local to the AMHS Management Domain operating the AFTN/AMHS Gateway. For a better interpretation of a given error situation at the control position, the subject information object may be sent in conjunction with the error reported to the control position.

6.2.3.2.5.5 For some categories of error situations the SARPs specify the actions to be taken, e.g. message rejection and generation of an appropriate service message (to the AFTN) or non-delivery report (to the AMHS). The specified actions aim at minimizing the assistance of the control position. However it may be a matter of policy local to the

AMHS Management Domain operating an AFTN/AMHS Gateway to try to reduce the occurrence of message rejection with the assistance of the control position.

6.2.3.2.6 **Interface between the ATN Component and the Message Transfer and Control Unit**

6.2.3.2.6.1 The exchange of information at the interface between the ATN Component and the Message Transfer and Control Unit is made using Transfer Envelopes, i.e. an interface to the Message Transfer service.

6.2.3.2.6.2 Other specifications are possible for an MHS AU. In particular, the selection of a Submission/Delivery interface would also have been possible. The reason to select a transfer interface is the possibility which is then given to the Message Transfer and Control Unit to generate non-delivery reports (NDRs) and delivery-reports. Such a possibility would not have been available if a submission/delivery interface had been selected. The ability to generate NDRs is considered particularly useful for the mapping certain AFTN service messages, i.e. those which indicate that the specified message recipient is unknown.

6.2.3.2.6.3 In terms of implementation, a message transfer API (which also allows to transfer reports) may then be used between the ATN Component and the Message Transfer and Control Unit when developing an AFTN/AMHS Gateway.

6.2.3.2.6.4 Flow control mechanisms may be implemented in both directions between the gateway components, e.g. to ensure that no messages in excess are passed to the ATN Component when it is unable to transfer them to the ATS Message Server or AFTN/AMHS Gateway to which it is connected. Such mechanisms, including the triggering criteria, are an implementation matter which is out of the scope of the SARPs.

6.2.3.2.7 **Interface between the AFTN Component and the Message Transfer and Control Unit**

6.2.3.2.7.1 Likewise, the Message Transfer and Control Unit has the possibility in an AFTN/AMHS Gateway to generate AFTN Service Messages with end-to-end significance and to pass them over to the AFTN Component, upon receipt of a NDR indicating an unknown recipient specification in a subject message.

6.2.3.2.7.2 As mentioned above, flow control mechanisms may also be implemented in both directions between these gateway components, e.g. to ensure that no messages in excess are passed to the AFTN Component when it is unable to transfer them to the AFTN Centre to which it is connected. Such mechanisms, including the triggering criteria, are an implementation matter which is out of the scope of the SARPs.

6.2.3.3 *Traffic Logging in an AFTN/AMHS Gateway*

6.2.3.3.1 **Implementation and use of information**

In general, the way in which the specified information is logged is an implementation matter. The way in which the logged information is retrieved and used is an implementation or operational matter, respectively. Therefore such aspects are out of the scope of the SARPs.

6.2.3.3.2 **AFTN Component traffic logging**

6.2.3.3.2.1 This function is where the behaviour of the AFTN Component differs from that of an AFTN Station.

6.2.3.3.2.2 Upon reception of a message from the AFTN, the AFTN Component behaves as an AFTN Station.

6.2.3.3.2.3 Upon generation of an AFTN message in the AFTN Component, long-term retention of the message in its entirety is performed in the AFTN Component. This may only happen for service messages, since otherwise the AFTN Component is not supposed to generate any message.

6.2.3.3.2.4 Messages received by the AFTN Component from the Message Transfer and Control Unit do not need to be logged in their entirety since the AFTN Component is not the initial originator of the message. Therefore, in this case, the logging requirement placed on the AFTN Component is equivalent to that of an AFTN Centre, that is to retain only the message heading, address and origin parts, and the action taken thereon.

6.2.3.3.3 **ATN Component traffic logging**

The traffic logging to be performed in the ATN Component of an AFTN/AMHS Gateway is equivalent to that of an ATS Message Server.

6.2.3.3.4 **Message Transfer and Control Unit traffic logging**

6.2.3.3.4.1 The main goal of the logging to be performed in the Message Transfer and Control Unit is to keep track of all information objects which have passed through the Message Transfer and Control Unit, and in particular to be able to identify the relationship between e.g. a received AMHS message and the converted AFTN message, for traceability purposes.

6.2.3.3.4.2 In case of duplication of information with either the traffic log of the ATN Component or of the AFTN Component, there is no requirement to implement different logs, provided that adequate mechanisms are implemented to allow the use of these traffic logs by the Message Transfer and Control Unit or in relation with the Message Transfer and Control Unit.

6.2.3.3.4.3 The nature of the information which is logged (and the way in which it is logged) in case of error situations in the Message Transfer and Control Unit is an implementation matter which depends on the way such situations are handled on a local basis.

6.2.3.3.5 **Relationship between these traffic logs**

In implementation terms, it is not necessary to implement three different logs in an AFTN/AMHS Gateway. In case of duplication of information between the Message Transfer and Control Unit traffic log and either the traffic log of the ATN Component or of the AFTN Component, it is only necessary that adequate mechanisms are implemented to allow the use of these traffic logs by the Message Transfer and Control Unit.

6.2.3.4 ***Address conversion in an AFTN/AMHS Gateway***

6.2.3.4.1 An AF-address may be converted in two different manners in an AFTN/AMHS Gateway:

a) if a corresponding MF-address including any combination of O/R address attributes has been allocated to the user identified by the AF-address, then a mapping process using fully configured look-up tables is necessary;

b) if no such address has been allocated, then the default conversion process for such an AF-address in an AFTN/AMHS Gateway (or more specifically in the Message Transfer and Control Unit of an AFTN/AMHS Gateway) aims at converting the AF-address into an XF-address, by means of a partly algorithmic method.

6.2.3.4.2 Case b) may occur for both indirect or direct users. In such a case, a look-up table is still necessary to identify the address attributes of the AMHS Management Domain to which the user belongs.

6.2.3.4.3 The term look-up table is used above and in the SARPs to describe in a simple manner the role of the function. However, many solutions can be envisaged when implementing such a function. For example, it is not necessary to implement two different look-up tables for the two mapping processes, a) and b) identified above. Also other types of data structures, e.g. relational databases, may be used to implement the required function.

6.2.3.5 ***Conversion functions of an AFTN/AMHS Gateway***

6.2.3.5.1 **General view of conversion functions**

6.2.3.5.1.1 These functions are performed by the Message Transfer and Control Unit. In the SARPs, the specification of these functions is split in accordance with the flow direction through the AFTN/AMHS Gateway. The conversion function in the direction from AFTN to AMHS is specified in section 3.1.2.3.4 of the SARPs and the conversion function in the direction from AMHS to AFTN is specified in section 3.1.2.3.5 of the SARPs.

6.2.3.5.1.2 The entire set of information objects processed by an AFTN/AMHS Gateway, together with the section of the SARPs where the relevant processing is specified, is depicted in Figure 6.2-8.

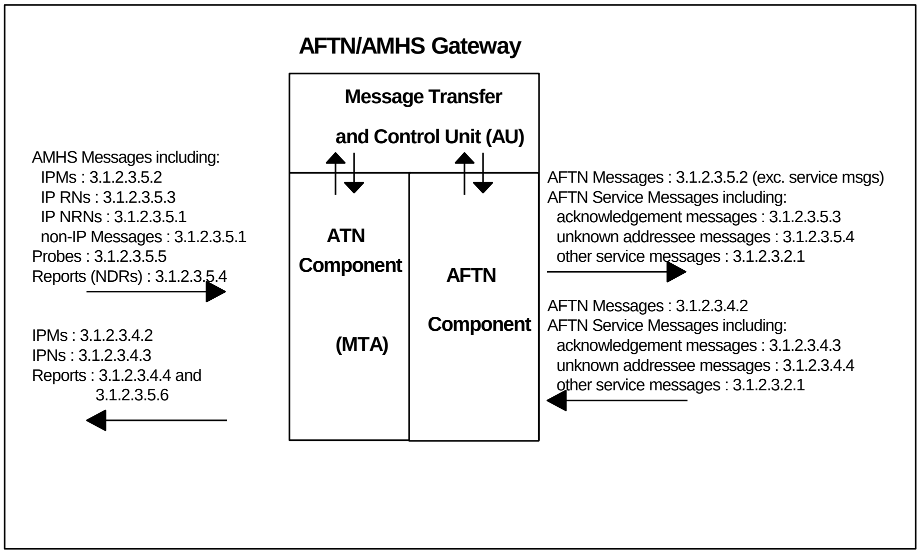

AFTN/AMHS Gateway
Message Transfer
and Control Unit (AU)
ATN
Component
(MTA)
AFTN
Component
AMHS Messages including:
IPMs : 3.1.2.3.5.2
IP RNs : 3.1.2.3.5.3
IP NRNs : 3.1.2.3.5.1
non-IP Messages : 3.1.2.3.5.1
Probes : 3.1.2.3.5.5
Reports (NDRs) : 3.1.2.3.5.4
IPMs : 3.1.2.3.4.2
IPNs : 3.1.2.3.4.3
Reports : 3.1.2.3.4.4 and
3.1.2.3.5.6
AFTN Messages : 3.1.2.3.5.2 (exc. service msgs)
AFTN Service Messages including:
acknowledgement messages : 3.1.2.3.5.3
unknown addressee messages : 3.1.2.3.5.4
other service messages : 3.1.2.3.2.1
AFTN Messages : 3.1.2.3.4.2
AFTN Service Messages including:
acknowledgement messages : 3.1.2.3.4.3
unknown addressee messages : 3.1.2.3.4.4
other service messages : 3.1.2.3.2.1

Figure 6.2-8. Information objects processed by the gateway

6.2.3.5.2 Scenarios for the AFTN/AMHS gateway operation

6.2.3.5.2.1 Elementary scenarios

6.2.3.5.2.1.1 An elementary scenario is a scenario describing the gateway behaviour upon receipt of a single information object.

6.2.3.5.2.1.2 Depending on the direction of the considered traffic flow and on the nature of the received information object, the different elementary scenarios which may occur at an AFTN/AMHS Gateway are depicted starting from Figure 6.2-9 and up to Figure 6.2-16.

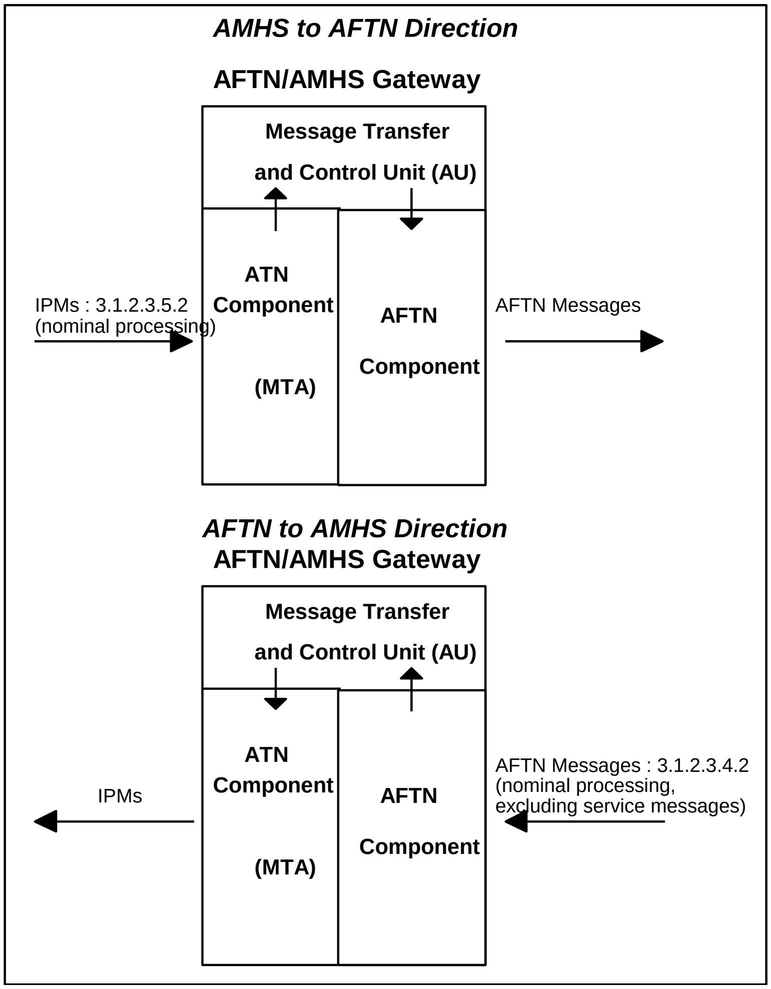


Figure 6.2-9. Conversion between an AMHS IPM and an AFTN Message

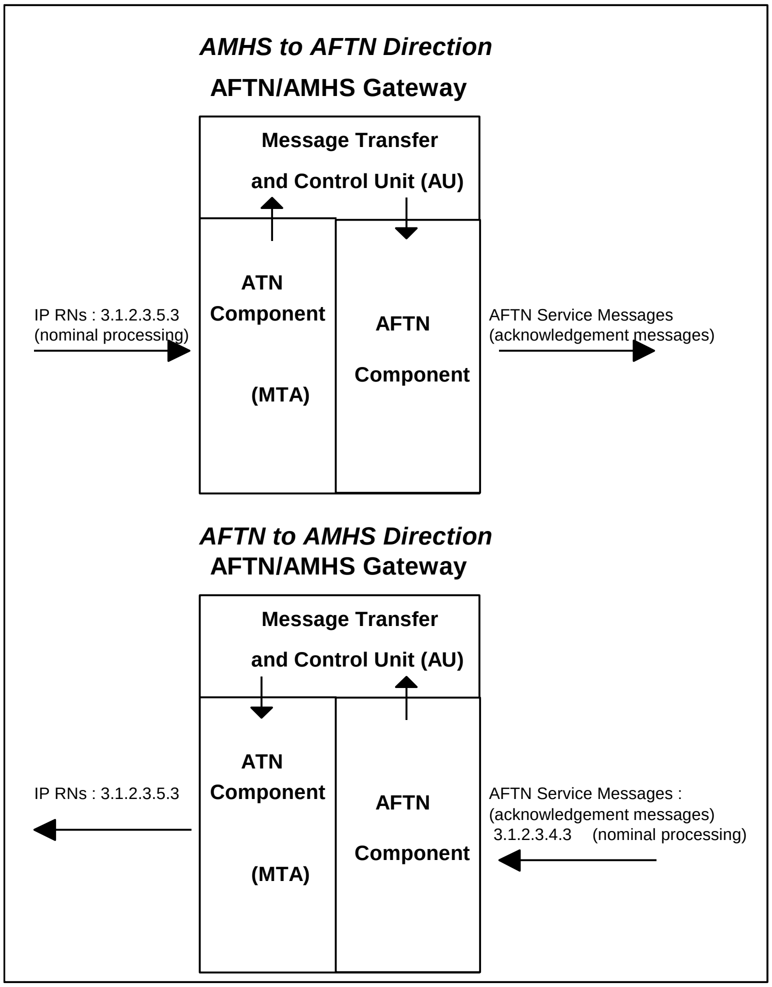


Figure 6.2-10. Conversion between an AMHS IP RN and an AFTN acknowledgement message

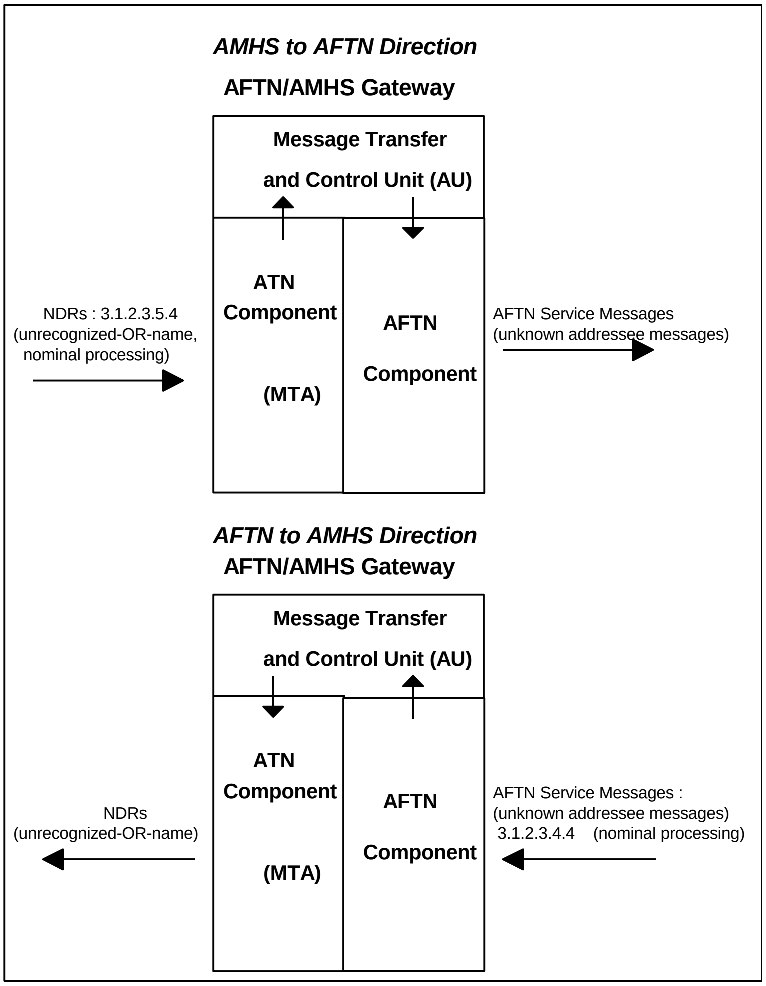


Figure 6.2-11. Conversion between an AMHS NDR (unrecognized O/R name) and an AFTN unknown addressee service message

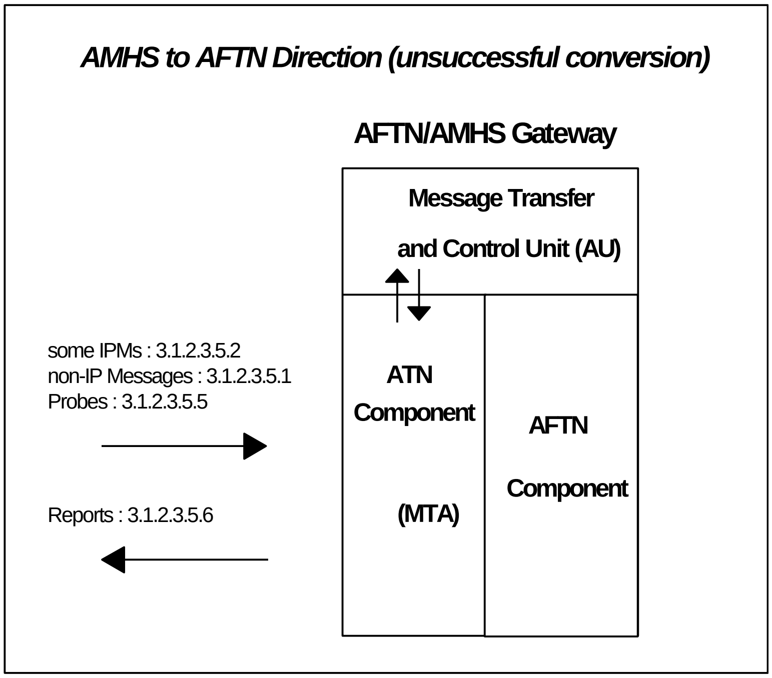


Figure 6.2-12. Unsuccessful conversion of incoming AMHS information objects in the MTCU

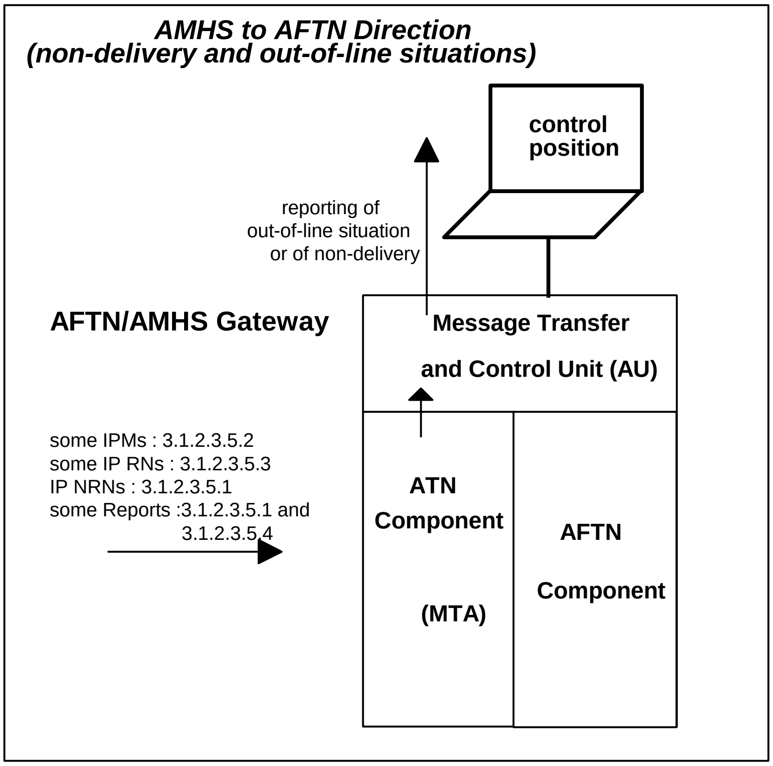


Figure 6.2-13. AMHS non-delivery and out-of-line situations

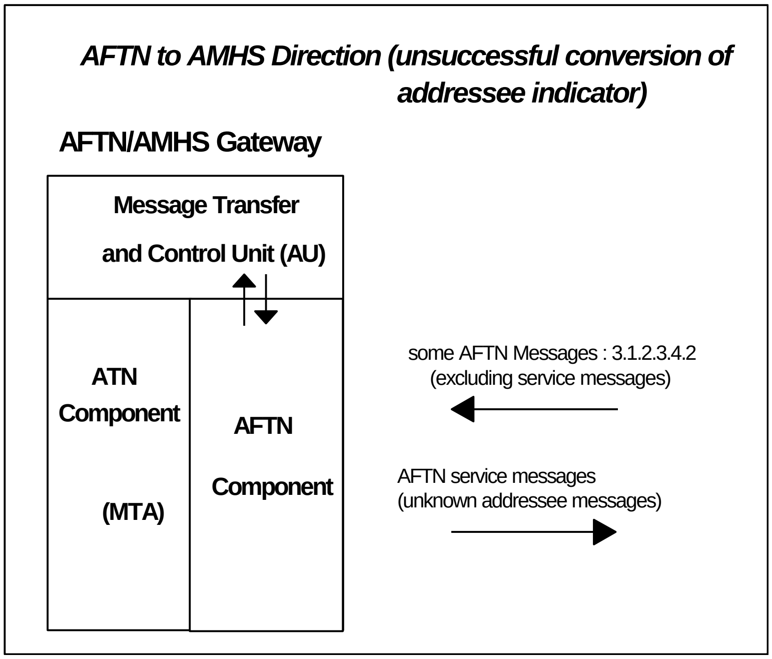

AFTN to AMHS Direction (unsuccessful conversion of addressee indicator)
AFTN/AMHS Gateway
Message Transfer and Control Unit (AU)
ATN Component (MTA)
AFTN Component
some AFTN Messages : 3.1.2.3.4.2 (excluding service messages)
AFTN service messages (unknown addressee messages)

Figure 6.2-14. Unsuccessful conversion of addressee indicator in incoming AFTN message

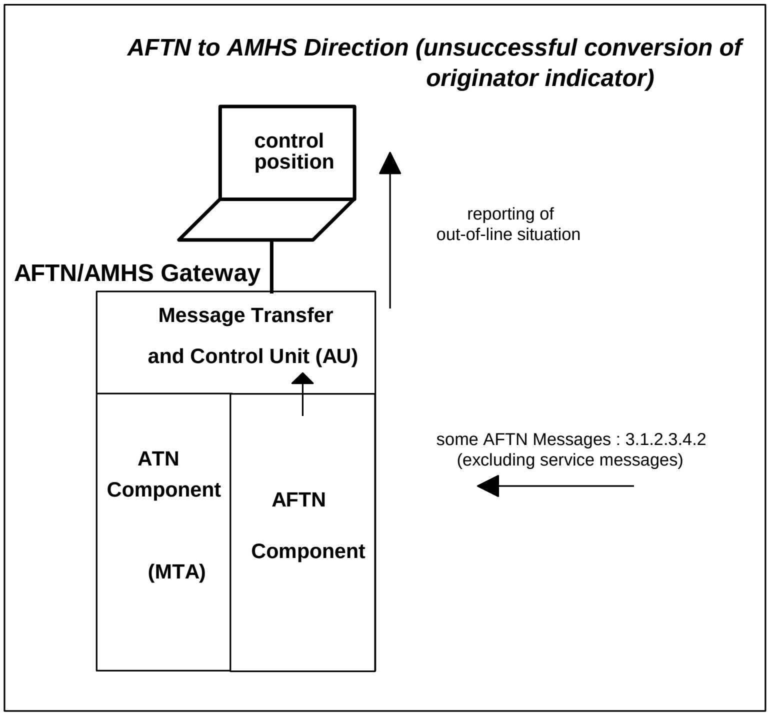

AFTN to AMHS Direction (unsuccessful conversion of originator indicator)
control position
reporting of out-of-line situation
AFTN/AMHS Gateway
Message Transfer and Control Unit (AU)
ATN Component (MTA)
AFTN Component
some AFTN Messages : 3.1.2.3.4.2 (excluding service messages)

Figure 6.2-15. Unsuccessful conversion of originator indicator in incoming AFTN message

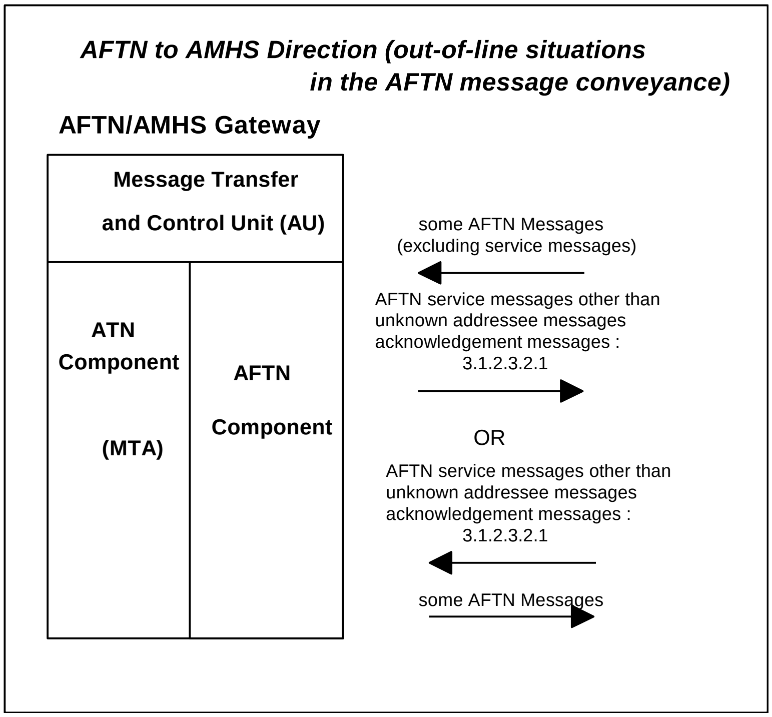

AFTN to AMHS Direction (out-of-line situations in the AFTN message conveyance)
AFTN/AMHS Gateway
Message Transfer and Control Unit (AU)
ATN Component (MTA)
AFTN Component
some AFTN Messages (excluding service messages)
AFTN service messages other than unknown addressee messages acknowledgement messages : 3.1.2.3.2.1
OR
AFTN service messages other than unknown addressee messages acknowledgement messages : 3.1.2.3.2.1
some AFTN Messages

Figure 6.2-16. Out-of-line situations in relation with incoming AFTN messages

6.2.3.5.2.2 **Combined scenarios**

6.2.3.5.2.2.1 A combined scenario is a scenario describing the gateway behaviour in cases where the gateway is involved several times in an overall information exchange, due to the use of AFTN service messages, and AMHS reports or notifications.

6.2.3.5.2.2.2 The scenarios described hereafter include only nominal processing by the gateway. They address the following cases:

a) acknowledgement of SS-priority messages; and

b) message rejection due to the use of an unknown addressee indicator or recipient O/R address.

6.2.3.5.2.2.3 They are illustrated by Figure 6.2-17 and Figure 6.2-18, respectively.

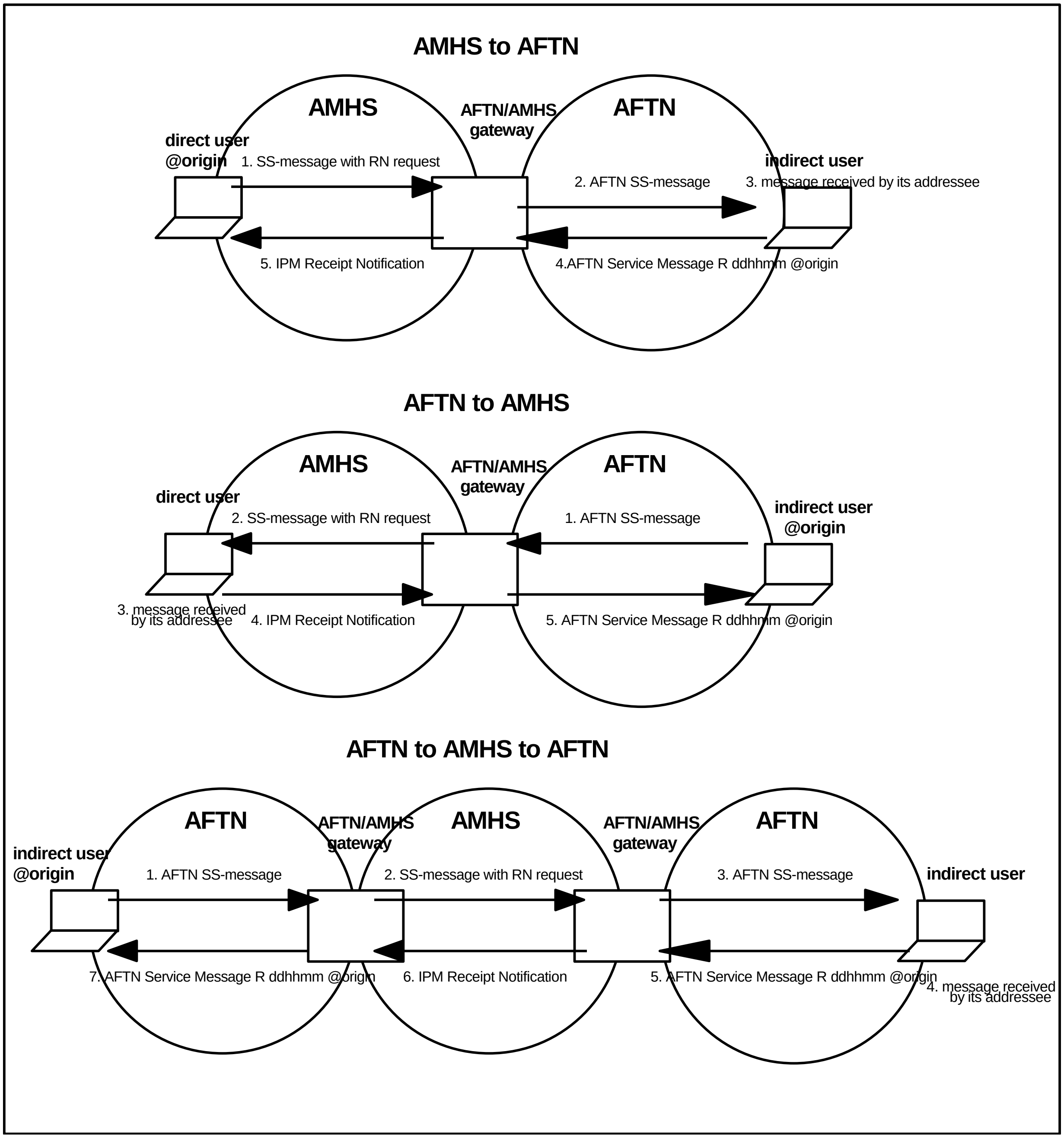

AMHS to AFTN
AMHS
AFTN/AMHS gateway
AFTN
direct user @origin
1. SS-message with RN request
indirect user
2. AFTN SS-message
3. message received by its addressee
5. IPM Receipt Notification
4.AFTN Service Message R ddhhmm @origin
AFTN to AMHS
AMHS
AFTN/AMHS gateway
AFTN
direct user
2. SS-message with RN request
1. AFTN SS-message
indirect user @origin
3. message received by its addressee
4. IPM Receipt Notification
5. AFTN Service Message R ddhhmm @origin
AFTN to AMHS to AFTN
AFTN
AFTN/AMHS gateway
AMHS
AFTN/AMHS gateway
AFTN
indirect user @origin
1. AFTN SS-message
2. SS-message with RN request
3. AFTN SS-message
indirect user
7. AFTN Service Message R ddhhmm @origin
6. IPM Receipt Notification
5. AFTN Service Message R ddhhmm @origin
4. message received by its addressee

Figure 6.2-17. Acknowledgement of SS-priority messages

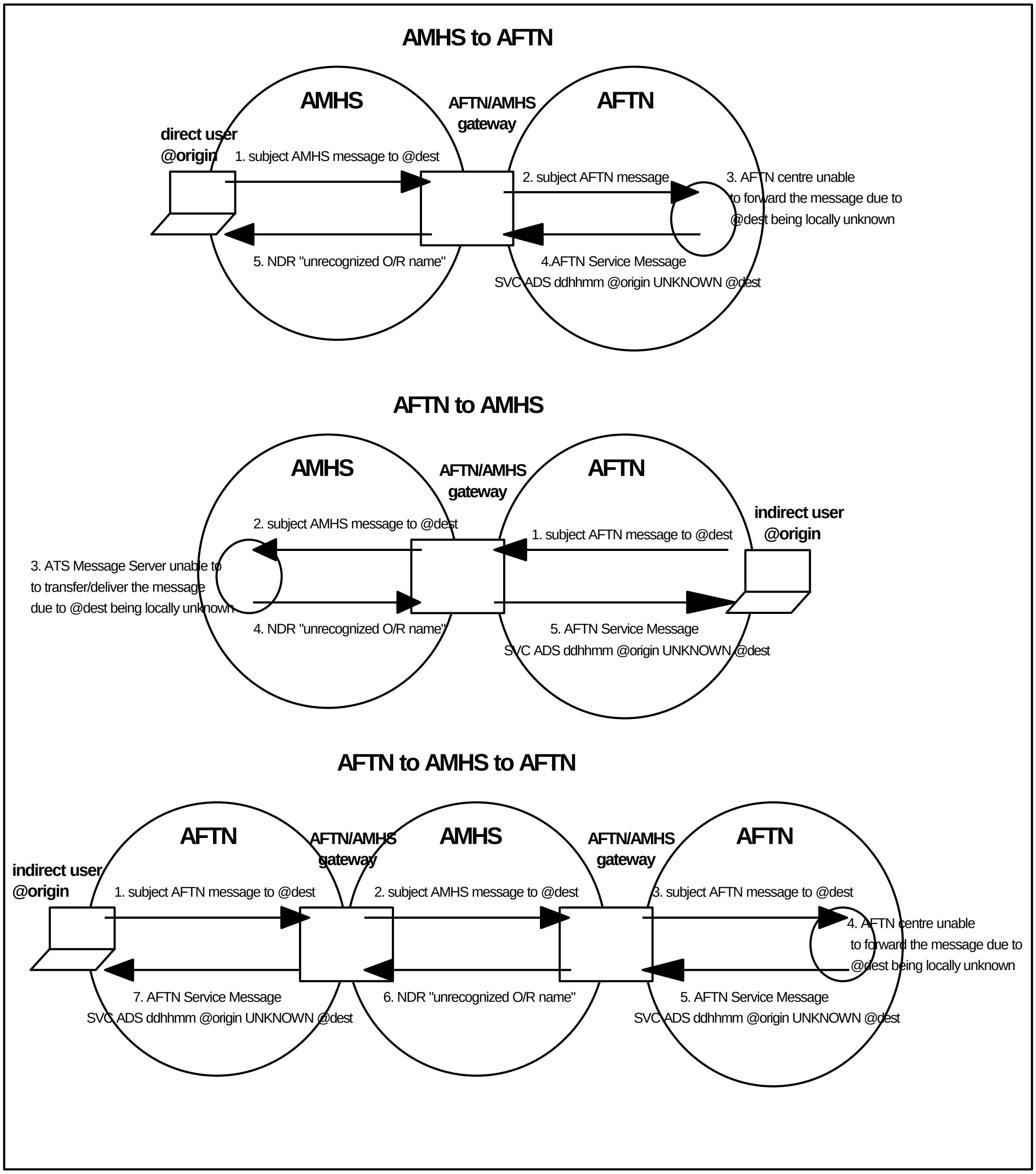

AMHS to AFTN
AMHS
AFTN/AMHS gateway
AFTN
direct user @origin
1. subject AMHS message to @dest
2. subject AFTN message
3. AFTN centre unable to forward the message due to @dest being locally unknown
4.AFTN Service Message SVC ADS ddhhmm @origin UNKNOWN @dest
5. NDR "unrecognized O/R name"
AFTN to AMHS
AMHS
AFTN/AMHS gateway
AFTN
indirect user @origin
1. subject AFTN message to @dest
2. subject AMHS message to @dest
3. ATS Message Server unable to to transfer/deliver the message due to @dest being locally unknown
4. NDR "unrecognized O/R name"
5. AFTN Service Message SVC ADS ddhhmm @origin UNKNOWN @dest
AFTN to AMHS to AFTN
AFTN
AFTN/AMHS gateway
AMHS
AFTN/AMHS gateway
AFTN
indirect user @origin
1. subject AFTN message to @dest
2. subject AMHS message to @dest
3. subject AFTN message to @dest
4. AFTN centre unable to forward the message due to @dest being locally unknown
5. AFTN Service Message SVC ADS ddhhmm @origin UNKNOWN @dest
6. NDR "unrecognized O/R name"
7. AFTN Service Message SVC ADS ddhhmm @origin UNKNOWN @dest

Figure 6.2-18. Message rejection due to the use of an unknown addressee indicator or recipient O/R address

6.2.3.5.3 AFTN to AMHS Conversion

6.2.3.5.3.1 Converted information objects

6.2.3.5.3.1.1 In this direction, the way to process the AFTN message may be determined from the contents of the first line of the message text. This first line refers to the string of characters included between the first character in a message text and the first CARRIAGE RETURN found therein.

6.2.3.5.3.1.2 An acknowledgement message (an AFTN service message acknowledging another AFTN message, then called "subject message" is characterized by its text which includes exclusively " R ddhhmm AFADDRES", where ddmmhh is a filing time as defined in Annex 10, Vol. II, 4.4.16.2.2.1 and AFADDRES is an AF-address.

6.2.3.5.3.1.3 An AFTN service message indicating that an addressee indicator in the subject message is unknown is characterized by its text which includes "SVC ADS ddhhmm AFADDRES", where ddmmhh is a filing time as defined in Annex 10, Vol. II, 4.4.16.2.2.1 and AFADDRES is an AF-address.

6.2.3.5.3.1.4 All other AFTN service messages are handled in the AFTN component only. In the particular case of AFTN service messages requesting correction by the originator of a message received mutilated, such messages are handled on the basis of a local specification, since no automated process can be specified due to:

a) the absence of an equivalent message in the MHS base standards. In effect, message mutilation, if it occurs in the AMHS, is automatically detected during the conveyance and reacted upon by means of MHS protocols; there is thus no need to request repetition from the originator, and

b) the difficulty to determine in an automated manner whether the AFTN/AMHS Gateway is in possession of an unmutilated copy of the message.

6.2.3.5.3.2 Error situations

6.2.3.5.3.2.1 Error situations may be reported for further actions to the control position in the following cases, classified in relation with the type of AFTN message received at the gateway:

a) "general" AFTN messages (excluding service messages)

   1) in case of conversion failure (general) in the MTCU or in case of transfer failure between the MTCU and the other components (section 3.1.2.3.4.1.3 of the SARPs);

   2) if the originator indicator of an AFTN message cannot be converted into an MF-Address (nor into an XF-Address) (section 3.1.2.3.4.2.1.4.1 of the SARPs);

b) AFTN acknowledgement service messages

1) if the AFTN acknowledgement message refers to a subject AFTN message which has not passed through the AFTN/AMHS Gateway (sections 3.1.2.3.4.3.1.1, 3.1.2.3.4.4.1.1 of the SARPs);

2) if the AFTN acknowledgement message refers to an IPM received without RN request;

c) AFTN unknown addressee service messages

1) if the AFTN unknown addressee message refers to a subject AFTN message which has not passed through the AFTN/AMHS Gateway (sections 3.1.2.3.4.3.1.1, 3.1.2.3.4.4.1.1 of the SARPs);

2) if, in the AFTN unknown addressee message, the unknown addressee indicator cannot be determined or mapped into a MF-Address (sections 3.1.2.3.4.4.1.2, 3.1.2.3.4.4.1.3 of the SARPs);

3) if the AFTN unknown addressee message is relative to a subject message which already caused the generation of a delivery-report by the AFTN/AMHS Gateway (section 3.1.2.3.4.4.1.4 of the SARPs).

d) AFTN message repetition service messages

1) if the AFTN message repetition message refers to a subject AFTN message which has not passed through the AFTN/AMHS Gateway (sections 3.1.2.3.4.3.1.1, 3.1.2.3.4.4.1.1 of the SARPs);

2) if the AFTN message repetition message refers to a subject AFTN message of which an unmutilated copy is not available at the gateway (sections 3.1.2.3.4.3.1.1, 3.1.2.3.4.4.1.1 of the SARPs).

6.2.3.5.3.2.2 In each of these cases, guidance may be given about the actions to be undertaken at the control position. Possible actions include:

a) the manual correction of the considered AFTN message before passing the message again for conversion to the MTCU;

b) the generation of an IPM carrying AFTN service information. This is requested by the SARPs in some of the cases listed above, but it may also be performed in other situations where this is not mandated by the SARPs. This action may only be performed when the out-of-line situation relates to an AFTN service message;

c) the generation of an AFTN message requesting repetition of the AFTN message being considered This action may only be performed when the out-of-line situation relates to an AFTN service message;

d) the transfer of conversion to another gateway. This action may only be performed when the out-of-line situation relates to an AFTN service message. The AFTN service message is then manually redirected to the gateway which initially converted the subject message, if this gateway and its AFTN address can be determined.

6.2.3.5.3.2.3 Furthermore, in certain cases, these actions should be undertaken only after appropriate actions have been performed to resolve the out-of-line situation.

6.2.3.5.3.2.4 The options available at the control position in each of the error situations identified above are summarized in Table 6.2-2.

Table 6.2-2. Actions at the control position upon receipt of AFTN service messages

	Options available at control position				
category of considered AFTN message / out-of-line situation	corrective action requested at control position to:	AND /OR	one of the three options below		
			manual message correction	generation of "service" IPM	repetition request to AFTN originator
"general" AFTN message					
unspecified conversion failure	correct conversion failure cause	and	yes		
transfer failure between gateway components	correct transfer failure cause				
failure of originator address conversion	check address mapping tables	and/or	yes		yes
AFTN acknowledgement message					
GW did not forward subject message	cancel re-routing reason	and		yes (SARPs)	
no-RN request	advise originator of subject message on need to request RN	and		yes (SARPs)	
AFTN unknown addressee message					

	Options available at control position				
category of considered AFTN message / out-of-line situation	corrective action requested at control position to:	AND /OR	one of the three options below		
			manual message correction	generation of “service” IPM	repetition request to AFTN originator
GW did not forward subject message	cancel re-routing reason	and		yes (SARPs)	
unknown addressee(s) invalid				yes (SARPs)	
failure of unknown addressee address conversion	check address mapping tables	and		yes (SARPs)	
delivery report already sent				yes (SARPs)	
AFTN request repetition message					
GW did not forward subject message	cancel re-routing reason	and		yes	
unmutilated copy of subject message not available				yes	

6.2.3.5.4 **AMHS to AFTN Conversion**

6.2.3.5.4.1 **Converted Information objects**

6.2.3.5.4.1.1 The processing applied to a received AMHS information object by the Message Transfer and Control Unit is one of the following, depending on the category of information object (message, probe or report) and content-type (interpersonal messaging, other):

a) process the object for conversion, or at least for further testing aiming at the determination of the gateway ability to convert the object based on envelope or contents parameter values;

b) rejection of the object, and generation of a non-delivery report; or

c) discard the message and report of an error situation. Such an event cannot normally happen under normal operating circumstances.

6.2.3.5.4.1.2 As indicated in the AFTN/AMHS Gateway overview, it is the role of the gateway to isolate the AFTN from any AMHS information object which has no AFTN equivalent. Therefore, the gateway behaviour must be specified for any standard MHS information object, since it may be received at the gateway.

6.2.3.5.4.1.3 This means that the gateway must be able to react upon reception of the following information objects:

a) messages, which content-type is built-in of any value (IPM-84 or IPM-88, EDI, voice or unidentified);

b) messages, which content-type is externally defined (external, specified by means of an Object Identifier);

c) probes, which content-type parameter has any of the aforementioned values;

d) reports, either delivery-reports or non-delivery-reports.

6.2.3.5.4.1.4 In the Basic ATS Message Service, only the IPM content-type (interpersonal-messaging-1984 or interpersonal-messaging-1988) is supported. Thus messages of any other built-in content-type, either unidentified (0), edi-messaging (35), or voice-messaging (40), or of any externally defined content-type, are rejected by an AFTN/AMHS Gateway.

6.2.3.5.4.1.5 Delivery reports are sent by the Message Transfer and Control Unit to the Control Position for further processing, since their reception is considered to be an out-of-line situation. This is due to the fact that a Message Transfer and Control Unit requests non-delivery-reports, but never delivery-reports when generating AMHS messages.

6.2.3.5.4.1.6 For IP messages, many different body-part types are defined in the base standards. These include body-parts defined as “basic”, and others defined as “extended”. Extended body-part types are Externally Defined, however some of them are defined in the base standards. Since, at some point in time, an IPM with any of these body-part types may reach an AMHS/AFTN Gateway, it is necessary to define the gateway behaviour upon receipt of an IPM with any kind of body-part defined in the base standards. It is likely that the reception of certain body-part types will not occur, since an IPM which body-part is, for example videotex, is not supposed to reach the gateway for onward transmission in the AFTN. Thus, the receipt of such an information object is an out-of-line situation which should happen only mistakenly. Furthermore, only some UA implementations are capable of generating IPMs with some of the defined body-part types. However, the AFTN/AMHS Gateway specification must be comprehensive with respect to the base standards, and the gateway specification in the SARPs includes provisions related to all defined body part types.

6.2.3.5.4.1.7 For IP Messages, the only body part types supported by the Message Transfer and Control Unit in reception are the following:

a) basic ia5-text;

b) extended ia5-text-body-part; and

c) extended general-text-body-part, for ISO 646 [9] and ISO 8859-1 [11] General Text Character Repertoires.

6.2.3.5.4.1.8 Body part types resulting from IPM forwarding, i.e. basic message and extended message body part, are not supported in reception by the AFTN/AMHS Gateway. This means that if such messages reach the gateway, they will be rejected and a NDR will be generated.

6.2.3.5.4.1.9 For this reason, if the information contained in an IPM is received by a direct user, who wishes to forward it to an AFTN user through an AFTN/AMHS gateway, it is necessary to use other mechanisms than the IPM forwarding as defined in the base standards. Such mechanisms are a local matter, they are based on the manual or automated creation of a new ia5-text or general-text message. The body of this new message may be constructed using the body of the received IPM (or DATA component of the body in case of general-text), and information about the originator of the initial IPM, if desired.

6.2.3.5.4.2 **Behaviour upon receipt of non-delivery reports**

6.2.3.5.4.2.1 When a non-delivery report (NDR) is received from the AMHS at an AFTN/AMHS Gateway, this report cannot be forwarded in the AFTN unless its non-delivery-diagnostic-code is "unrecognised-OR-name" (see section 3.1.2.3.5.1.3 of the SARPs). This may happen in particular for a subject message which was initially converted by the gateway. Thus, the NDR is passed to the control position for appropriate action. This scenario is depicted in Figure 6.2-19.

6.2.3.5.4.2.2 Under circumstances where routing is not symmetrical, i.e. where reports about a subject message are not routed along the same path in the opposite direction as the subject message itself, a NDR may be also be received by an AFTN/AMHS Gateway which did

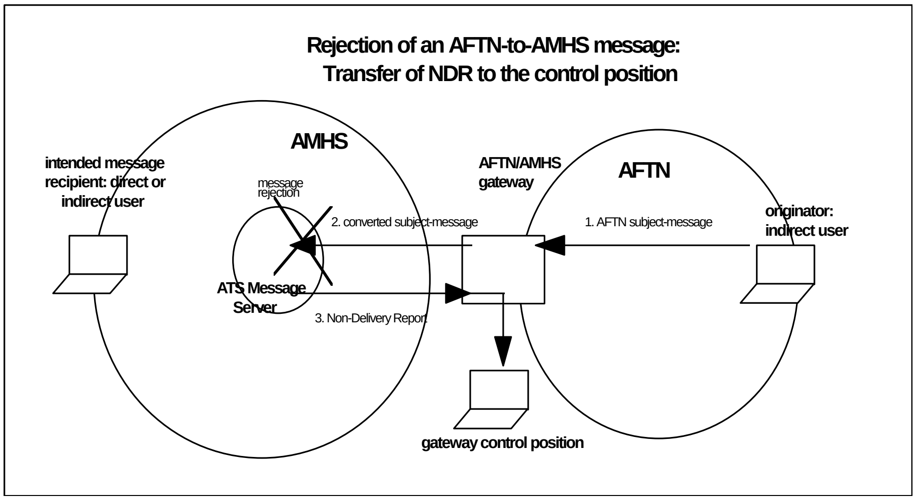

Rejection of an AFTN-to-AMHS message:
Transfer of NDR to the control position
AMHS
AFTN
intended message recipient: direct or indirect user
message rejection
AFTN/AMHS gateway
2. converted subject-message
1. AFTN subject-message
originator: indirect user
ATS Message Server
3. Non-Delivery Report
gateway control position

Figure 6.2-19. Rejection of an AFTN-to-AMHS message: Transfer of NDR to the control position

not convert the subject message. This section is general in nature and also addresses such a scenario, however some specific actions at the control position may be required in such a case.

6.2.3.5.4.2.3 A NDR includes two parameters giving indications about the causes of the non-delivery of the subject message. These are:

a) the non-delivery-reason-code (NDRC) element, which is mandatory in a NDR. It may take 8 different values;

b) the non-delivery-diagnostic-code (NDDC) element, which is optional in a NDR. It may take 48 different values. If present, it further refines the reason given in the NDRC.

6.2.3.5.4.2.4 The base standards specify the meaning of each abstract-value of these parameters. For some of these values (but not all) the base standards also specify, in the description of the MTA procedures, which value to use in which rejection circumstances.

6.2.3.5.4.2.5 The action undertaken at the control position should be based on the values of these parameters in the received NDR. In summary, the non-delivery of the subject message may result from three main types of situations:

a) an out-of-line situation in the AMHS, e.g. MTS congestion, or unability to transfer due to a MTA failure duration exceeding the repetition conditions at the previous MTA;

b) an out-of-line situation due to the subject message itself; or

c) a mismatch between the subject message and the capabilities of the intended recipient, e.g. if the message recipient does not support the encoded-information-types (EITs) of the subject message.

6.2.3.5.4.2.6 Situations b) and c) above are not supposed to occur under normal circumstances, for a subject message sent from an AFTN/AMHS Gateway to an ATS Message User Agent. Such situations are normally prevented to happen if both systems comply with the SARPs. However, guidelines may be given for behaviour at the control position, in case this would happen as the result of an out-of-line situation.

6.2.3.5.4.2.7 Potential actions are among the following, they may be combined as appropriate:

a) if the NDR indicates an error condition in the AMHS, undertake the appropriate action for the correction of the error, or check that the out-of-line situation has been resolved;

b) if the subject message originator is a person (as opposed to a computer application), send an AFTN " free text message" to the originator informing him that the subject message could not be delivered. A proposed text for this message is "DELIVERY

OF MESSAGE WITH ORIGIN filing_time originator_indicator FAILED WITHIN AMHS”,

c) request subject message repetition to the message originator, by means of an AFTN service message (“SVC QTA RPT...”), or repeat subject message from traffic log, if available.

6.2.3.5.4.2.8 If the subject message had not been converted initially by the AFTN/AMHS Gateway which received the NDR, it cannot be guaranteed that this gateway will be able to convert automatically the MF-Address of the NDR recipient, i.e. the MF-Address of the subject message originator, into the corresponding AF-Address. In such a case, either actions b) and c) above are excluded, or a preliminary investigation is necessary to determine manually the AF-Address.

6.2.3.5.4.2.9 Actions which should be preferred for each category of non-delivery causes are indicated in Table 6.2-3.

Table 6.2-3. Actions at the control position upon receipt of AMHS NDRs

non-delivery cause	action at control position		
	undertake action for error correction / check resolution	send AFTN free text message to originator	request subject message repetition to originator or repeat message if available
out-of-line situation in AMHS	yes		yes (should be performed only after correction of error)
out-of-line situation in subject message		yes	
mismatch between subject message and intended recipient capabilities	yes (see Note 1)	yes (see Note 1)	yes (should be performed only after correction of error)

Note 1.— One out of both, depending on mismatch type

6.2.3.5.4.3 Error situations

6.2.3.5.4.3.1 Inventory of error situations

6.2.3.5.4.3.1.1 Error situations may be reported for further actions to the control position in the following cases:

a) if a non-receipt-notification (NRN) is received (section 3.1.2.3.5.1.2 of the SARPs);

b) if the received information object is not among the objects to be converted, nor within the objects to be explicitly rejected (section 3.1.2.3.5.1.5 of the SARPs, last instance rejection);

c) if a message is received whose priority-indicator in the ATS-Message-Header is "SS" and which does not request a RN (section 3.1.2.3.5.2.3.3 of the SARPs);

d) if a RN is received relative to a subject IPM whose priority-indicator in the ATS-Message-Header differs from "SS" (section 3.1.2.3.5.3.1.2 of the SARPs); and

e) if a RN is received relative to a subject IPM which had not been generated by the AFTN/AMHS Gateway (section 3.1.2.3.5.3.1.1 of the SARPs).

6.2.3.5.4.3.1.2 In each of these cases, guidance may be given about the actions to be undertaken at the control position.

6.2.3.5.4.3.2 Receipt of NRN

6.2.3.5.4.3.2.1 The reception of a NRN usually indicates that the originator of the NRN, i.e. one of the intended recipients of the subject IPM, did not receive the subject IPM although the message was delivered to him. The NRN conveys a parameter which is the Non-receipt Reason, optionally supplemented with two other parameters, which are the Discard Reason and the Auto-forward Comment. These parameters refine the explanation given by the non-receipt reason.

6.2.3.5.4.3.2.2 Depending on the values of these parameters, possible actions at the control position are as follows:

a) ignore the NRN, for example if the auto-forward comment indicates that the subject IPM has been forwarded to an eventual recipient which replaces the intended recipient;

b) handle the NRN as if it were a NDR, in accordance with the guidance given in section 6.2.3.5.4.2;

c) undertake an appropriate action to avoid that other non-receipts occur for the same recipient.

6.2.3.5.4.3.3 **Receipt of information objects which cannot be converted**

6.2.3.5.4.3.3.1 The receipt of an information object which is not among the objects to be converted, nor within the objects to be explicitly rejected, is clearly an error of which the repetition should be avoided.

6.2.3.5.4.3.3.2 Two types of action may be undertaken at the control position:

a) inform the message originator, by means of an IPM conveying service information, that the information object received at the gateway cannot be conveyed in the AFTN and has consequently been received mistakenly at the gateway;

b) undertake appropriate action to prevent, by policy and/or technical means (e.g. routing, delivery capabilities, etc.) such information objects from being conveyed to the gateway or to the MTCU in the gateway.

6.2.3.5.4.3.4 **Receipt of an SS priority message non-compliant with AMHS receipt notification request rules**

6.2.3.5.4.3.4.1 This situation corresponds to case c) under section 6.2.3.5.4.3.1.1. It means that the message parameters contradict the rules specified in the SARPs and explained in section 6.2.2.2.3 (Use of priority-indicators and notification-requests).

6.2.3.5.4.3.4.2 Since the SS priority message does not request a RN, the SARPs specify that the AFTN/AMHS Gateway must send back a “service IPM” to the message originator, upon receipt of the AFTN acknowledgement service message for the SS priority message. This service IPM contains the text of the AFTN acknowledgement service message.

6.2.3.5.4.3.4.3 The receipt of such a message is already an indication to the originator, that the specification was not complied with. However, an additional action by the control position is possible, consisting in recalling explicitly to the message originator that an SS priority message should be sent with a RN-request.

6.2.3.5.4.3.5 **Receipt of a RN relative to a non-SS priority message**

6.2.3.5.4.3.5.1 This situation corresponds to case d) under section 6.2.3.5.4.3.1.1. The receipt of a RN for a non SS priority message means that the subject message has been generated by another AFTN/AMHS Gateway, mistakenly requesting a receipt notification, although the message has a priority different from SS.

6.2.3.5.4.3.5.2 This leads to the interpretation that:

a) the gateway which generated the subject message does not comply with the SARPs;

b) routing in the AMHS was not symmetrical at the moment of conveying the subject message and the considered RN.

6.2.3.5.4.3.5.3 Under such circumstances, the following actions can be undertaken at the control position of the gateway which received the RN:

a) discard the RN; and/or

b) advise the control position of the AFTN/AMHS gateway which originated the subject message about the detected uncompliance. This may require a manual investigation to determine which gateway is concerned.

6.2.3.5.4.3.6 Receipt of a RN relative to a subject IPM generated by another gateway

6.2.3.5.4.3.6.1 This situation corresponds to case e) under section 6.2.3.5.4.3.1.1.

6.2.3.5.4.3.6.2 The SARPs specify that the AFTN/AMHS Gateway must return a NDR to the RN originator.

6.2.3.5.4.3.6.3 Under such circumstances, the following actions can be additionally undertaken at the control position of the gateway which received the RN:

a) ignore the RN, and leave it to the recipient of the subject message (RN originator) to acknowledge the receipt of the subject message; or

b) advise the control position of the AFTN/AMHS gateway which originated the subject message. This may require a manual investigation to determine which gateway is concerned.

6.3 ATN Pass-Through Service

6.3.1 Introduction

When implementing the AFTN/ATN Type A Gateway, consideration should be given in creating separate code modules for each of the three components specified in the SARPs:

a) AFTN component,

b) ATN component, and

c) Message Transfer and Control Unit (MTCU).

6.3.2 AFTN Component

6.3.2.1 *General role*

6.3.2.1.1 The AFTN component of the AFTN/ATN Type A Gateway is the part of the gateway which interfaces with the AFTN. The purpose of the AFTN component is to isolate all AFTN specific functions in a common module or code section.

6.3.2.1.2 In implementing the AFTN component, consideration should be given to implement the gateway in a manner that allows the AFTN component to be implemented and tested separately from the remainder of the gateway.

6.3.2.2 *Message Retention*

6.3.2.2.1 There are two types of message retention (or logging) that are required by the SARPs:

a) short term; and

b) long term.

6.3.2.2.2 Short term retention must be accomplished, e.g. by maintaining the message in memory, until it can be ascertained that the message has been received by the remote AFTN centre or station, or until a timer expires. This timer must last at least one hour, it is started upon transmission of the message. Once responsibility for the message has been assumed by another system, the gateway needs no longer maintain the message in the short term log.

6.3.2.2.3 Long term retention requires that the AFTN component store the message in a file or some other long-term storage. The simplest method of achieving this requirement is to immediately copy all messages to a file as soon as they are received (as the first function in processing). The writing of the file and the maintenance of the file will satisfy the retention requirements.

6.3.2.2.4 When implementing the logging of messages, the implementor should consider the format of the log file so that searching, retrieval, and display of logged messages can be performed. Further, specific requirements for logging information, such as message receipt time-stamps, should also be considered.

6.3.2.3 *Addressing*

6.3.2.3.1 The AFTN component is responsible for analyzing AFTN addresses and determining the appropriate AFTN centre for forwarding, for messages received from the MTCU and destined for the AFTN. In the simplest case, the AFTN component is connected to a single AFTN centre and all AFTN messages go over that link. In a more complicated situation, the AFTN addresses will need to be analyzed and compared against a table which maps AFTN addresses to corresponding links.

6.3.2.3.2 The analysis must include the capability of supporting the full AFTN addressing structure such as a maximum of three AFTN address lines.

6.3.3 **ATN Component**

6.3.3.1 *General role*

6.3.3.1.1 The ATN component is the interface between the MTCU and the ATN. It is designed as a single service primitive which provides the communication with the MTCU. The ATN component makes use of the ATN upper layer communication services as defined in Sub-Volume 4 of the SARPs.

6.3.3.2 *GA-Data request service primitive*

6.3.3.2.1 The GA-Data request primitive is used to pass information from the MTCU to the ATN. The primitive contains parameters which provides the necessary addressing and QoS information needed by the ATN.

6.3.3.2.2 The user data parameter is used to pass a complete AFTN message in the IA5 character set.

6.3.3.2.3 The called and calling addresses are required and consist of the 8 character facility designation.

6.3.3.2.4 The priority parameter may be included by the MTCU and it may only contain the priority of the AFTN message as determined during AFTN message processing.

6.3.3.2.5 The ATN component uses the information found in the parameters and maps the information into the parameters of the D-Start request. The SARPs specify a Control Function (CF) which defines the mapping of the GA-Data request parameters to the D-Start request parameters.

6.3.3.3 *CF*

The SARPs specify that if a dialogue exists between two gateways, then it should be used for the passing of data. This assumes that dialogues between gateways will be maintained for a period of time after it is established, even if there is no data. It also assumes that the existence of a dialogue is known globally within the gateway when new messages from the MTCU are passed to the CF. An implementation may choose to release a dialogue as soon as it finishes passing a single message. This would mean that for each AFTN message passed between two gateways, a new dialogue would be established.

6.3.4 **MTCU**

6.3.4.1 *General role*

The MTCU performs the central task of mapping from AFTN to ATN and ATN to AFTN.

6.3.4.2 *Message Logging*

The SARPs specify that the MTCU must log all messages. This can be combined with the message retention requirements specified for the AFTN component and does not mean that separate logs are required.

6.3.4.3 ***Address Mapping***

6.3.4.3.1 The central feature of the MTCU is the address mapping from one addressing plan to the other.

6.3.4.3.2 To map from the AFTN addresses to ATN addresses, the MTCU must:

a) analyse the AFTN message to determine the AFTN address; parse the AFTN address which may consist of one to three address lines and find an AFTN addressee indicator; look-up that addressee indicator in the mapping table and find the facility designation of the ATN end system which will be the receiver; repeat this step this step for each AFTN addressee indicator;

b) for each ATN end system, make a copy of the AFTN message with the stripped address line and send to the appropriate number of ATN end systems.

6.3.4.3.3 To map from the ATN address to the AFTN addresses nothing needs to be done. The address of the destination(s) is located in the AFTN message and is handled transparently by the AFTN component.

6.3.5 Exception handling

6.3.5.1 ***General***

When implementing a gateway, consideration should be given to the types of errors that can occur and how those errors should be processed.

6.3.5.2 ***Messages arriving from AFTN***

Errors may be detected in the format of the message, e.g., an invalid address line. If that is the case, the message may be logged into an error file and a manual process may be employed to correct the error. Alternatively, the message could be rejected and the originator notified accordingly.

6.3.5.3 ***Messages to be sent to the AFTN***

The only error is a failure in transmission. The most appropriate action is to log the message to an error file and to process manually. Alternatively, an automatic resend could be implemented on the automatic processing of the error log.

6.3.5.4 ***Messages arriving from the ATN***

There are no error conditions since the processing of the AFTN message is handled in the MTCU.

6.3.5.5 ***Messages being sent to the ATN***

Errors can occur if the ATN is inoperative or the destination end-system is unavailable. In these cases, the message should be logged to an error file and either manually or automatically resent.

6.3.5.6 ***MTCU processing***

Errors can occur in the addressing stripping and address mapping. In these cases, the message should be logged in an error log and manually modified and resent. For invalid messages, a service message is sent back the AFTN station as specified in the SARPs.

7. *ICC APPLICATION*

7.1 Introduction

7.1.1 Identification

7.1.1.1 This document is Part III, Chapter 7 of the Comprehensive ATN Manual

7.1.2 Purpose

7.1.2.1 In line with normal ICAO practice, this document was developed as a companion document to the Aeronautical Telecommunication Network (ATN) Air Traffic Services (ATS) Interfacility Data Communication (AIDC) Standards and Recommended Practices (SARPs). It may be read alongside the AIDC SARPs, in order to provide a greater understanding of the specification itself, or it may be read instead of the AIDC SARPs by readers who simply want to understand the purpose of the AIDC Application rather than the detail of the specification.

7.1.3 Scope

7.1.3.1 This document provides guidance material for those implementing the AIDC Application.

7.1.3.2 This document does not define any mandatory or optional requirements for AIDC, nor does it define any recommended practices. This document does not instruct users on how to use the AIDC application in a particular operational environment.

7.1.4 Background

7.1.4.1 ATS today use a number of applications that provide specific services and require ground-ground communications. A significant number of these applications are still based on a manual ATS environment and, in general, use the asynchronous transmission procedures and technology of the Aeronautical Fixed Service (AFS) to communicate.

7.1.4.2 With the introduction of the Future Air Navigation Systems (FANS) and ATN concepts by ICAO and the subsequent push for the implementation of an automated global ATS environment, many of the current services provided by ATS are to be assessed to determine how these services may be provided using ATN technologies.

7.1.4.3 The focus for this assessment is the Automatic Dependent Surveillance (ADS) Panel. Operational requirements developed by the ADS Panel are laid out in the ICAO *Manual of Air Traffic Services Data Link Applications* (Doc 9694) [7]. This document is aligned with the text of [7] as approved by the Air Navigation Commission in December 1997.

7.1.4.4 The initial set of operational services that have been reviewed to date are the following:

a) Flight Notification,

b) Flight Coordination,

c) Transfer of Executive Control

d) Transfer of Communications,

e) Transfer of Surveillance Data, and

f) Transfer of General Information (flight-related data or free text messages).

7.1.4.5 Additional high-level operational services that are to be reviewed are as follows:

a) Flight Planning,

b) Airspace Management,

c) Air Traffic Flow Management,

d) Aeronautical Information, and

e) Meteorological Information.

7.1.5 Inter-Centre Communication SARPs

7.1.5.1 The Inter-Centre Communication (ICC) SARPs will define a set of ATN applications making use of a generic communications stack that will support the operational services identified in 7.1.4.4 and 7.1.4.5 above.

7.1.5.2 It is anticipated that each of these ATN applications will be either a pure ground-ground application or a combined ground-ground and air-ground application.

7.1.6 ICC SARPs for the AIDC Application

7.1.6.1 The first of the applications developed for ICC is the AIDC application.

7.1.6.2 The AIDC application enables the exchange of information between ATS Units (ATSUs) for support of critical Air Traffic Control (ATC) functions, such as notification of flights approaching a Flight Information Region (FIR) boundary, coordination of boundary conditions and transfer of control and communications authority.

7.1.6.3 The AIDC application supports the following operational services:

a) Flight Notification,

b) Flight Coordination,

c) Transfer of Executive Control,

d) Transfer of Communications,

e) Transfer of Surveillance Data (Surveillance reports from conventional radars), and

f) Transfer of General Information (free text messages).

7.1.6.4 AIDC is strictly an ATC application for exchanging tactical control information between ATS Units. It does not support the exchange of information with other offices or facilities.

7.1.6.5 In the nature of ICAO SARPs, it is assumed that the two ATSUs are under the authority of two different States. However, a State may find it convenient to apply the AIDC SARPs for relevant communications between two ATSUs under its own authority.

7.1.6.6 For reasons related to their own operational environment, states might not be willing to incur the cost of developing and implementing the full AIDC application as defined in the SARPs. The conditions which must be met to ensure interoperability are specified in SARPs [2] 3.2.9.1. In addition, an AIDC application claiming conformance to the SARPs needs to implement a minimum amount of functionality specified in the SARPs. This minimum amount of functionality requires that each system implementing the AIDC application is able to:

a) generate, transmit, receive and process a common set of AIDC messages, depending on the local operational requirements — this may include: Notify, CoordinateInitial, CoordinateUpdate, and CoordinateCancel;

b) decode and encode the information contained in the non-optional data elements of these messages;

c) decode or ignore, as appropriate, the information contained in the optional data elements of these messages;

d) inform its AIDC-User of whether the exchange of a given message was succesfully completed.

7.1.7 Relationship to other ATN applications

7.1.7.1 AIDC is a ground-ground application. The AIDC SARPs do not specify any direct relationship with other ATN applications. The AIDC application is a stand-alone application which can be developed, certified, installed and operated completely independently from other ATN applications.

7.1.7.2 AIDC is envisaged as the first in a suite of ground-ground applications meeting the whole range of requirements for Inter-Centre Communications (ICC).

7.1.7.3 In a scenario where all ATN applications are implemented, an operating concept can be conceived, as follows, from the perspective of a single ATSU receiving, handling, and transferring a flight:

a) flight plan data is received by means of the ATS Message Service (AMHS) application;

b) notification, coordination and transfer of the flight from the transferring ATSU is performed through AIDC;

c) data link address and application information is exchanged with the aircraft through the Context Management (CM) application;

d) dialogue with the aircraft and aircrew is maintained through the Controller-Pilot Datalink Communication (CPDLC) application;

e) aircraft position and intent is monitored through the Automatic Dependent Surveillance (ADS) application;

f) notification, coordination and transfer of the flight to the next ATSU is performed through AIDC.

7.1.8 The application of this concept to several ATSUs responsible for neighbouring FIRs follows by considering such a scenario from the viewpoint of each ATSU separately.

7.1.9 **Structure**

a) 7.1 — Introduction — contains the reasons for providing guidance material. It also provides a brief overview of the AIDC application, and the relationship of the AIDC application with other ATN applications. It also identifies the applicable reference documents.

b) 7.2 — Overview of the AIDC application — describes generic concepts that are used throughout the AIDC SARPs and Guidance Material,

c) 7.3 — The AIDC Abstract Service — gives a functional breakdown of the various services that AIDC provides. It describes the peer-to-peer interaction, including reasons for why particular information is used or not used, and what operations on the information are expected,

d) 7.4 — Commentary on the AIDC SARPs — clarifies any functionality that was not addressed in the previous section on a chapter by chapter basis,

e) 7.5 — Dimension (size of messages) — gives some sample encoding sizes for guidance on what capacities need to be allowed for in order to meet normal operational expectations.,

f) 7.6 — Indexes and Tables — provides a cross-reference to the ASN.1 definitions used in the AIDC application.,

g) 7.7 — Example scenarios — gives some examples as to what typical scenarios one can expect in course of normal AIDC operation, and

h) 7.8 — Example encoding — outlines some actual sample encoding of typical AIDC messages.

7.1.10 **References**

7.1.10.1 References to other sections of this Comprehensive ATN Manual are given directly, with a qualifying Part number where necessary. References to sections of the Detailed Technical Provisions of the ATN SARPs [2] are noted as 'SARPs <section>' giving the section number. Other documents referenced in the text are indicated thus [n], where 'n' is the number of the reference in the list below.

[1] Annex 10 — *Aeronautical Telecommunications,* Volume III, Part I, Chapter 3

[2] *Manual of Technical Provisions for the ATN* (Doc 9705)

[3] Annex 10 — *Aeronautical Telecommunications*, Volume III, Part I, Chapter 3

[4] Annex 11 — *Air Traffic Services*

[5] *Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services* (PANS-RAC, Doc 4444)

[6] *Location Indicators* (Doc 7910)

[7] *Manual of Air Traffic Services Data Link Applications* (Doc 9694)

[8] *Report of the APANPIRG Asia/Pacific Regional ATS Inter-Facility Data Communications Interface Control Document Task Force Meeting*, Canberra, Australia, 9-13 January 1995, ICAO Asia and Pacific Office

[9] *North Atlantic Common Coordination Interface Control Document*, North Atlantic Systems Planning Group, ICAO European and North Atlantic Office, Version 1.2, 1994

[10] *EUROCONTROL Standard for On-Line Data Interchange*, Edition 2 (awaiting publication), EUROCONTROL, Brussels

[11] *ISO/IEC 7498-1:1994*. Information technology — Open Systems Interconnection — Basic Reference Model. Reference: ITU-T Rec. X.200 (1994)

[12] *ISO/IEC 7498-3:1989*. Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 3: Naming and Addressing. Reference: ITU-T Rec. X.650 (1992)

[13] *ISO/IEC 8649:1994*. Information processing systems — Open Systems Interconnection — Service definition for the Association Control Service Element (second edition). Reference: ITU-T Rec. X.217 (1992)

[14] *ISO/IEC 8649/PDAM 1:1995*. Information technology — Open Systems Interconnection — Service definition for the Association Control Service Element — Amendment 1: Fast associate mechanism

[15] *ISO/IEC 8822:1994*. Information technology — Open Systems Interconnection — Presentation service definition (Second edition). Reference: ITU-T Rec. X.216 (1994)

[16] *ISO/IEC 8822/DAM 2:1994*. Information technology — Open Systems Interconnection — Presentation service definition — Amendment 1: efficiency enhancements. Reference: ITU-T Rec. X.216 Addendum 1 (1995)

[17] *ISO/IEC 8824-1:1994*. Information Technology -OSI Abstract Syntax Notation One (ASN.1). — Part 1: Specification of basic notation. Reference: ITU-T Rec. X.682 (1994)

[18] *ISO/IEC 8824-1/Amd.1:1995*. Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1) -Part 1: Specification of Basic Notation -Amendment 1: Rules for Extensibility

[19] *ISO/IEC 8824-3:1995*. Information technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN. 1) — Part 3: Constraint specification

[20] *ISO/IEC 8825-1:1995*. Information technology — Specification of ASN.1 encoding rules — Part 1: Specification of Basic Encoding Rules (BER) Reference: ITU-T Rec. X.691 (1993)

[21] *ISO/IEC 8825-2:1996*. Information technology — Open Systems Interconnection — Specification of Encoding Rules for Abstract Syntax Notation One (ASN.1) — Part 2: Packed encoding rules. Reference: ITU-T Rec. X.691 (1995)

[22] *ISO/IEC 9545:1994*. Information technology — Open Systems Interconnection — Application Layer structure (second edition)

[23] *ISO/IEC 10731:1994*. Information technology — Open Systems Interconnection — Conventions for the definition of OSI services. Reference: ITU-T Rec. X.210 (1993)

7.2 Overview of the AIDC Application

7.2.1 Introduction

7.2.1.1 The ATN AIDC application specified in 3.2 of [2] is based on the concepts and requirements specified in the ICAO *Manual of Air Traffic Services Data Link Applications* (Doc 9694) (Part VI of [7]), developed by the ICAO ADSP. In addition, the ADSP has specified that the AIDC application should conform to the ATN protocols for its data link operations.

7.2.1.2 The basic concepts embodied in the AIDC application are ATSU roles and phases of coordination. The AIDC application implements information exchanges that are defined for each role and phase.

7.2.1.3 It has been ensured that the development of the operational concepts by the ADSP and the development by the ATNP of the technical means of achieving them keep in step with each other. However, the ADSP is generally looking at a longer timescale than the current ATNP initial implementation programme, and this will inevitably mean that some elements of its work have not been incorporated into the SARPs.

7.2.1.4 In particular several messages defined in Part VI of [7] are not implemented in the version of the AIDC application described in this chapter. These messages are those that support coordination between more than two ATSUs: CoordinateReady, CoordinateCommit, and CoordinateRollback.

7.2.2 Roles of an ATSU

7.2.2.1 An ATSU can fulfil various roles in relation to a flight from the perspective of AIDC:

a) *Controlling ATSU* (C-ATSU) when it is the ATS Unit currently responsible for control of a flight;

b) *Downstream ATSU* (D-ATSU) when it is responsible for control of an airspace region through or near which the aircraft will progress at some future time;

c) *Receiving ATSU* (R-ATSU) when it is about to assume control responsibility for the flight.

7.2.2.2 The notations *ATSU1* and ATSU2 are also used in [2] and [7] to refer to two separate ATSUs when it is not necessary to identify a specific role related to a flight.

7.2.2.3 At any time, an ATSU may fulfil different roles in relation to each flight that it is handling.

7.2.3 Dialogues

7.2.3.1 A dialogue is a sequence of one or more related message exchange sequences. All exchanges of AIDC messages between ATSUs may be viewed as dialogues. AIDC dialogues support the following functions:

a) Notification;

b) Coordination;

c) Transfer;

d) General Information Interchange, and

e) Surveillance Data Transfer.

7.2.3.2 Most dialogues consist of a single primitive request and response. However, coordination and transfer dialogues involve multiple primitive sequences. The structure and sequence of AIDC dialogues is discussed further in 7.3 below.

7.2.3.3 An AIDC dialogue concerns a single flight. An ATSU can only maintain one AIDC dialogue per flight with another ATSU.

7.2.3.4 A dialogue is typically initiated when one of several events, such as the following, occurs:

a) proximity to an airspace boundary;

b) controller action;

c) a change in a flight's cleared trajectory affecting a D-ATSU; or

d) surveillance data update.

7.2.4 Phases of Coordination

7.2.4.1 Each flight, as it proceeds from one FIR to another, goes through three phases of coordination, listed in sequential order as follows:

a) Notify Phase, in which the trajectory and any changes may be conveyed to a D-ATSU from the C-ATSU prior to coordination using a Notification dialogue;

b) Coordinate Phase, in which the flight's trajectory is coordinated between a C-ATSU and one or more D-ATSUs when the flight approaches a common boundary using a Coordination dialogue; and

c) Transfer Phase, in which communications and executive control authority is transferred from the C-ATSU to the R-ATSU using a Transfer dialogue.

7.2.4.2 ***Notification***

7.2.4.2.1 Once a flight is airborne, the trajectory may be changed by the C-ATSU. The trajectory and any changes may be sent to affected D-ATSUs using a Notification dialogue. A flight does not actually have to enter an FIR for notification to be relevant: it need only pass within the lateral separation minima associated with an FIR. Notification occurs in advance of coordination with a D-ATSU.

7.2.4.3 ***Coordination***

7.2.4.3.1 The coordination phase begins when the flight is a parameter time or distance from a common boundary between FIRs controlled by two or more ATSUs. A flight does not actually have to enter an FIR for coordination to be required: it need only pass within the lateral separation minima associated with an FIR. The cleared flight trajectory and boundary crossing conditions (if the flight actually crosses the boundary) are negotiated between the ATSUs using an initial Coordination dialogue until mutually agreed upon conditions are obtained. The re-Coordination dialogue(s) provides a mechanism for modifying an existing coordination, and may be invoked while the aircraft remains in the border areas of two or more FIRs.

7.2.4.4 ***Transfer***

7.2.4.4.1 Transfers of communications and control authority normally occur when the flight is about to enter a new FIR. However, these may occur at any point subsequent to initial coordination subject to approval of both ATSUs. Transfer dialogues are employed to perform these functions.

7.2.4.5 ***Regimes***

7.2.4.5.1 Each of the above phases is represented in the AIDC application by a corresponding regime: Notifying, Coordinating, Transferring.

7.2.5 AIDC Service Primitives

7.2.5.1 The information exchanges (messages) specified in Part VI of [7] transferred by the AIDC application are defined in the AIDC SARPs [2] in terms of communications service primitives according to the accepted convention for specifying Open Systems Interconnection (OSI) communications functions [23].

7.2.5.2 A unique subset of AIDC service primitives is associated with each of the coordination regimes.

7.2.5.3 Other AIDC service primitives support ancillary ATC data exchanges between ATSUs, including:

a) transfer of surveillance data from radar reports, and

b) exchange of free-text messages.

7.2.6 The AIDC Functional Model

7.2.6.1 Figure 7.2-1 shows the AIDC Application as defined by the AIDC SARPs [2] in the context of a possible ATN implementation. The components shown model the AIDC application and its environment and do not have to be reflected in any specific implementation. In particular, the other ATN application-entities are shown purely as an example. Nor is there any requirement to implement a real interface in an implementation that corresponds to any of the service boundaries shown.

7.2.6.2 The functional elements identified in this model are the following:

a) the AIDC-User, which represents the operational part of the AIDC system;

b) the AIDC Application Entity (AIDC-AE), which is the component defined and specified in the AIDC SARPs [2], comprising:

1) the AIDC Control Function (CF),

2) the AIDC Application Service Entity (AIDC-ASE), and

3) the Association Control Service Element (ACSE), defined in [13];

c) the ATN Service Provider Interface, which represents the rest of the ATN.

7.2.6.3 *AIDC-User*

7.2.6.3.1 The AIDC-User makes use of AIDC using the communication service provided by the AIDC-AE through the AIDC-AE Service boundary.

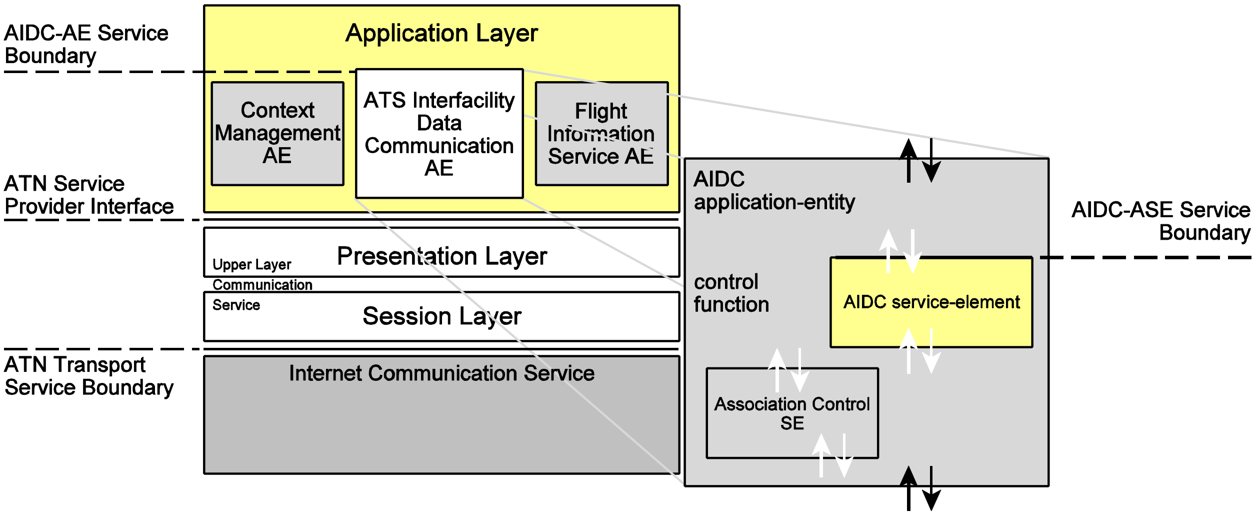

AIDC-AE Service Boundary
Application Layer
Context Management AE
ATS Interfacility Data Communication AE
Flight Information Service AE
ATN Service Provider Interface
Upper Layer Communication Service
Presentation Layer
Session Layer
ATN Transport Service Boundary
Internet Communication Service
AIDC application-entity
control function
AIDC service-element
Association Control SE
AIDC-ASE Service Boundary

Figure 7.2-1. Functional model of the AIDC Application

7.2.6.3.2 The structure of the AIDC-User is outside the scope of ATN SARPs. However, it can be imagined as representing the combination of ATC software and the human air traffic controller involved in handling a flight, including in particular a flight data processing subsystem, if present.

7.2.6.3.3 The involvement of the controller reflects the need for informed human oversight of automated decision-making. This principle of human-centred automation is described in Part I.2A of [7].

7.2.6.3.4 Agreement to participate in the AIDC Application by the AIDC-User implies its acceptance of certain constraints on its behaviour, which are specified in SARPs 3.2.9 [2].

7.2.6.4 *AIDC-AE*

7.2.6.4.1 The specification of the AIDC-AE follows the structure laid down in the OSI Upper Layer Architecture [22], as adapted to the ATN requirements in the SARPs (ULCS) [2].

7.2.6.4.2 Following this approach, the AIDC-AE is fully specified by specifying its constituent ASEs and the AIDC-AE CF.

7.2.6.4.3 Readers are encouraged to consult the ATN SARPs 4 (ULCS) [2] and Part IV.2 of this Manual for detailed rationale for, and explanation of, the ATN Upper Layer Architecture, and for detailed implementation guidance. The present chapter provides an explanation of the AIDC-AE structure that is independent of the ULCS SARPs and related guidance material. For the reader who has assimilated the ULCS documentation, the relation of the AIDC application structure to that specified in the ULCS is explained.

7.2.6.5 *AIDC-ASE*

7.2.6.5.1 The AIDC-ASE is the element in the communication system which executes the AIDC-specific protocol, enforcing the AIDC specific primitive sequencing actions, timer management, and error handling.

7.2.6.5.2 The AIDC-ASE can be seen as having two boundaries:

a) the upper service boundary comprises the primitives that may be submitted to or delivered by the AIDC-ASE;

b) the lower boundary comprises the primitives that may be submitted by or delivered to the AIDC-ASE.

7.2.6.6 *AIDC-AE CF*

7.2.6.6.1 The AIDC-AE CF is responsible for mapping primitives received from one element to another within the AIDC functional model. Thus it can be seen as a ‘glue’ that links the AIDC-ASE and the ACSE.

7.2.6.6.2 The CF is specified in terms of the possible events that can occur. These comprise:

a) submission of primitives by the AIDC-User (across the AIDC service boundary),

b) delivery of primitives by the ATN Service Provider (across the ATN Service Provider Interface),

c) delivery of primitives by the AIDC-ASE (across the upper AIDC-ASE service boundary),

d) delivery of primitives by the ACSE (across the upper ACSE service boundary), and

e) expiration of internal CF timers.

7.2.6.6.3 Each event results in further processing, which may include the submission or delivery of further service primitives as well as timer manipulations. The CF maintains an internal state on which the specific processing required for each event depends.

7.2.6.7 *ATN Service Provider Interface*

7.2.6.7.1 The ATN Service Provider Interface provides access to OSI layers 1-6 of the ATN supporting the AIDC application. Thus the Service Provider Interface is the presentation service boundary defined in [15].

7.2.6.7.2 The ATN Service Provider is specified in the ATN SARPs [2] for Upper Layer Communications Service (ULCS) (4) and Internet Communications Service (ICS) (5). The AIDC application uses the ATN ICS, via the ATN Service Provider Interface, to ensure that ATC information is exchanged in a reliable and timely manner between ATSUs.

7.2.6.8 *Implementation*

7.2.6.8.1 In implementing the AIDC application, the functionality described by the AIDC-AE model is to be implemented.

7.2.6.8.2 The software developed, whether it is a collection of modules, a single module, or a number of separate application processes, should, from the AIDC-User's point of view, exhibit the behaviours defined by the model described in the SARPs [2].

7.2.6.8.3 SARPs [2] requirements and recommendations on the AIDC-AE only need to be interpreted in terms of what they imply for information transferred over the ATN. Provided that the AIDC functionality is made available to the AIDC-User and the AIDC protocol, as observed at the interface between the real AIDC end system and the ATN subnetwork, conforms to the SARPs, the implementor is not constrained.

7.2.7 Dialogues and use of connections

7.2.7.1 The AIDC service hides the use of the underlying communications service from the AIDC-User. There is a direct correspondence between a dialogue and an underlying application association, such that information dealing with a single flight is transferred over a single application association.

7.2.7.2 Certain AIDC primitives (which include the necessary addressing information) implicitly establish an application-association. These can be considered as initiating a dialogue.

7.2.7.3 In version 1 of the AIDC SARPs [2], an application-association corresponds directly to a single presentation connection, session connection, and transport connection. In sizing an AIDC implementation, the maximum number of concurrent transport connections needs to be considered; this should be at least the maximum number of flights for which the ATSU may need to maintain dialogues with other ATSUs.

7.2.7.4 Version 1 of the AIDC application does not directly support coordination by a C-ATSU of a single flight with several D-ATSUs. When AIDC is being used to coordinate a single flight with several D-ATSUs, separate instances of the AIDC application with a corresponding dialogue exist for each pair of ATSUs together with their supporting connections.

7.2.8 User-confirmation primitive

7.2.8.1 Many of the AIDC dialogues include the User-confirmation primitive. This is used by a responding AIDC entity to indicate either:

a) where the primitive is one of a sequence in a dialogue, that a preceding primitive has been delivered and is being processed, or,

b) where the primitive is the last in a sequence, to acknowledge the end of a phase of the dialogue, or

c) where the primitive is isolated, to acknowledge its delivery.

7.2.9 Naming and addressing

7.2.9.1 *Naming*

7.2.9.1.1 The names used in the AIDC application are OSI application-entity-titles, as defined in [12].

7.2.9.1.2 The structure of the AIDC names is specified in the SARPs [2] for ULCS (4), based on the ICAO location indicators [6] of the respective ATSUs, as follows:

{iso (1) identified-organization (3) icao (27) atn-end-system-ground (2)
<end-system-id> (n) operational (0) idc (6)}

7.2.9.2 *Addressing*

7.2.9.2.1 The AIDC application name has to be resolved into the corresponding presentation address (which is effectively the ATN NSAP address). How this is done (the application-layer-directory-facility as defined in [11]) is a local matter: for instance, the mapping may be statically configured in the AIDC end system, or it may be achieved through a distributed directory application.

7.2.10 **Version handling**

7.2.10.1 The version of the SARPs [2] corresponding to this GM has the AIDC-AE version number set to 1.

7.2.10.2 The version number is transferred in the association establishment phase as part of the application-context-name, which is the following object-identifier defined in the ULCS SARPs [2]:

{iso (1) identified-organization (3) icao (27) atn-ac (3) version1 (1)}

7.2.10.3 An implementation that only supports the version 1 application-context will reject an establishment attempt specifying another version. It would be dangerous to do otherwise, because as a result of the use of the Packed Encoding Rules, transmitted encodings are liable to differ significantly according to the version, and the version 1 application cannot reliably decode the data transmitted by another version.

7.2.10.4 The AIDC abstract syntax is specified using extensibility markers for certain constructs that are considered candidates for future changes. Although a version 1 application should be able to handle messages whose abstract syntax differs only where extensibility markers existed in the initial specification, only the information elements defined in the initial specification will be decoded and acted upon. Consequently it may be assumed that the AIDC-AE version number will be incremented when the abstract syntax is changed, unless both of the following hold:

a) the modifications are extensions with respect to the extensibility markers, and

b) the implications of ignoring the information contained in such extensions are operationally acceptable.

7.2.10.5 An implementation that supports both version 1 and another version of the AIDC application is responsible for maintaining compatibility with implementations that only support version 1, where this is desired. For instance, an implementation supporting other versions of AIDC may detect the rejection of an establishment attempt using a version other than one and retry using the version 1 application-context.

7.2.10.6 Implementations using a full OSI upper-layer profile would be able to negotiate the application-context, using standard OSI upper-layer functionality. The FastByte upper-layer profile adopted in the ATN ULCS excludes this capability.

7.2.10.7 The AIDC service primitives, their format and sequencing rules may all be subject to change in future versions of AIDC. The AIDC-User may have to invoke different processing logic based on the selected AIDC version and so will need to be involved in any version negotiation. If an AIDC implementation is capable of negotiating between several supported versions of the AIDC application, implementors need to consider providing the following features in a software interface to the AIDC application that are not modelled in the AIDC abstract service:

a) a means for the AIDC application to report the version of the application that is in use following a dialogue-establishing primitive (these primitives are Notify, CoordinateStart, and InfoTransfer), and

b) a means by which the AIDC-User can control which of the supported versions the AIDC implementation uses.

7.2.11 Rationale for user/system functionality

7.2.11.1 The AIDC-AE supports the dialogues needed for notification, coordination and transfer, and enforces the necessary logical and temporal constraints on the primitives in each dialogue. Thus two valid implementations of the AIDC-AE will be able to interact, passing data to each other in the correct order; they will be able to check the format of the data, ensure that it has been sent with the appropriate service primitive and ensure that the peer ASE is behaving according to the requirements in the SARPs [2].

7.2.11.2 Within the AIDC-AE, the AIDC-ASE contains all the logic specific to the AIDC application, while the AIDC-CF links it to the supporting communications services (ACSE and presentation service).

7.2.11.3 Some of the requirements in the SARPs [2] apply to the AIDC-User. These provisions are required to ensure the interoperability of the two peers.

7.2.12 Relationship to existing ground-ground mechanisms

7.2.12.1 *Distribution of flight plan information*

7.2.12.1.1 Messages defined in [5] are widely used to transfer flight plan information between ATSUs, typically some hours in advance of the flight. These messages are currently often transferred using the Aeronautical Fixed Service (AFS) [3]. Other arrangements may be used in some circumstances; for instance, in Europe a central flight plan processing system receives and validates all flight plans and then distributes them to relevant ATSUs using a regionally defined protocol.

7.2.12.1.2 Distribution of flight plan information to relevant ATSUs is not essential for the use of AIDC. However, such information is typically needed in the ATSU for other reasons (eg, work scheduling, flow management). Availability of a flight plan in a D-ATSU reduces the controller’s workload and avoids the need to transfer large amounts of flight plan information in the Notification phase.

7.2.12.1.3 One objective of the Notification phase is to allow the D-ATSU to verify that it has a suitable flight plan for the flight concerned, and to obtain one if no flight plan is found.

7.2.12.1.4 The flight plan may have been forwarded by the originating centre, or by a central facility:

a) as discussed in 7.2.12.1.1, there are various means by which flight plans can be distributed ahead of the flight;

b) in addition, AIDC may be used to transfer flight plan information using the Notify or Info-transfer services.

7.2.12.1.5 In the last resort, it may be necessary to create a flight plan manually in the D-ATSU.

7.2.12.1.6 Assumptions and conditions relating to the provision of flight plans and the possible actions in the event that a D-ATSU cannot find a flight plan will need to be agreed between each pair of ATSUs that implement coordination and transfer supported by AIDC.

7.2.12.2 *Notification, coordination and transfer*

7.2.12.2.1 Message exchanges have been developed on a regional basis, based on the coordination messages specified in [5], to support the notification, coordination and transfer of flights (for example see [10], [9], [8])..

7.2.12.2.2 The requirements for the AIDC application were strongly influenced by the experience of developing and implementing these regional protocols. Use of AIDC as an alternative may provide benefits as ATN infrastructure becomes more widespread, and is envisaged in some of the relevant regional agreements.

7.2.12.2.3 As of 1998, these regional approaches are independent of ATN. The communication mechanism supporting message exchange is provided by special point-to-point protocols, or by the AFS. Use of the ATN ICS as an alternative transport mechanism may provide benefits.

7.2.13 Relationship to ATSMHS

7.2.13.1 There is no specific relationship between the AIDC application and the ATSMHS applications.

7.2.13.2 The ATN AMHS application ([2], 3.1.2) provides a possible means of distributing flight plan information to support the operation of AIDC.

7.2.14 Relationship to Air-Ground applications

7.2.14.1 There is no specific relationship between the AIDC application and the ATN air-ground applications.

7.2.14.2 The CPDLC application can be used by the C-ATSU to pass information and commands to the aircrew as needed to achieve the transfer of the flight. Use of CPDLC in combination with AIDC in this way may provide benefits in terms of reduced controller workload.

7.2.14.3 The ADS application can be used by the ATSUs concerned with the transfer of a flight to monitor the conformance of the aircraft to the agreed coordination conditions.

7.2.14.4 AIDC supports the transfer of independent or manual dependent surveillance information between ATSUs. The ADS report forwarding application (which is strictly a ground-ground application, but not necessarily between two ATSUs, though defined in combination with the ADS air-ground application) specified in SARPs 2.2.2 [2] supports transfer of surveillance information from automatic dependent surveillance.

7.3 The AIDC Abstract Service

7.3.1 Service conventions

7.3.1.1 The AIDC service defined in SARPs [2] 3.2.3, as an OSI service definition, is an abstract boundary in the end system. It follows the conventions for OSI services in [23].

7.3.1.2 There is no requirement to implement a real interface, such as a set of procedure calls, in an implementation corresponding to the AIDC service boundary.

7.3.1.3 However, the AIDC service boundary does represent a sensible point in an implementation where a major software interface could be defined (a better point, for instance, than the AIDC-ASE or presentation service boundaries). The AIDC service definition can be used as the basis for such a software interface, though the exact form of the interface would depend on the implementation technology. For instance, possible realisations of an AIDC software interface include:

a) messages sent by inter-process communication,

b) procedure calls,

c) software interrupts,

d) object method invocations.

7.3.1.4 The AIDC service as a whole is composed of 17 service primitives. Each primitive corresponds to a single element of communications functionality from the perspective of the AIDC-User and is defined in terms of:

a) its parameters (the information items provided by or to the AIDC-User, and their meaning), and

b) the allowed sequences of primitives.

7.3.1.5 A service primitive may occur in several forms:

a) request — the primitive is submitted by the AIDC-User;

b) indication — the primitive is delivered to the AIDC-User;

c) response — the primitive is submitted by an AIDC-User after processing a corresponding indication;

d) confirmation — the primitive is delivered to an AIDC-User in response to a preceding indication and response.

7.3.1.6 A confirmed primitive uses all of these forms, while an unconfirmed primitive uses at most request and indication.

7.3.1.7 The AIDC application has been designed using mainly unconfirmed primitives. In many cases the unconfirmed primitive is answered with a corresponding unconfirmed primitive that acknowledges it, as described in 7.2.8. The rationale for not defining these primitives as confirmed services is that the acknowledgement does not confirm the processing requested in the initial primitive; this confirmation (or otherwise) of processing is conveyed subsequently. In most cases there are several possible types of response and so it is more convenient in the specification not to merge the response types as the response and confirmation forms of the initial primitive, but rather to define them as separate unconfirmed primitives.

7.3.2 **Common services**

7.3.2.1 ***User-confirmation***

7.3.2.1.1 The User-confirmation service implements the AppAccept and AppError messages defined in Part VI.5 of [7].

7.3.2.1.2 The User-confirmation service is unconfirmed: ie, the use of the service consists of the submission of User-confirmation.request by one AIDC-User and the corresponding delivery of User-confirmation.indication to the peer AIDC-User.

7.3.2.1.3 The User-confirmation service is used in all regimes of the AIDC service, whenever an ‘Information’ parameter is delivered, to indicate whether or not the received parameter is valid according to the receiving AIDC-User.

7.3.2.1.4 A User-confirmation primitive may be

a) positive, indicated by the Result parameter having the abstract value ‘accepted’, or

b) negative, indicated by the Result parameter having the abstract value ‘rejected’.

7.3.2.1.5 A positive User-confirmation.indication does not signify acceptance of the information (eg, it does not accept a proposal of coordination conditions). It merely indicates that the information was ‘understood’ by the receiving AIDC-User (eg, the flight associated with proposed coordination conditions was recognised).

7.3.2.1.6 A negative User-confirmation signifies that the information was not understood.

7.3.2.1.7 The AIDC-AE enforces a time limit between the submission by an AIDC-User of a service primitive with an ‘Information’ parameter and the delivery of the corresponding User-confirmation.indication. If this limit is exceeded, the dialogue is aborted, resulting in the delivery of a Provider-abort.indication to both AIDC-Users with ProviderAbortReason having the abstract value ‘timerexpired’.

7.3.2.2 *End*

7.3.2.2.1 The End service implements the CoordinateCancel message defined in Part VI.5 of [7]. It may also be used outside the Coordination phase, as specified in the AIDC SARPs [2].

7.3.2.2.2 The End service is unconfirmed: ie, the use of the service consists of the submission of End.request by one AIDC-User and the corresponding delivery of End.indication to the peer AIDC-User.

7.3.2.2.3 The End service is used in all regimes to end a dialogue. The service can either terminate the dialogue (implicitly indicating successful completion of the dialogue), or cancel the dialogue.

7.3.2.2.4 Use of the End service to cancel a dialogue indicates that information, proposals, and/or agreements resulting from the dialogue are no longer valid. For instance, in a Coordination or Notification dialogue, the flight may have been re-routed under the control of the C-ATSU so that it will no longer enter the airspace controlled by the D-ATSU.

7.3.2.3 *User-abort*

7.3.2.3.1 The User-abort service does not correspond to any message defined in Part VI.5 of [7]. It is used by one AIDC-User to terminate the AIDC service.

7.3.2.3.2 The User-abort service is unconfirmed: ie, the use of the service consists of the submission of User-abort.request by one AIDC-User and the corresponding delivery of User-abort.indication to the peer AIDC-User.

7.3.2.3.3 As SARPs [2] 3.2.3.9.2 notes, the User-abort service may be invoked at any time for operational or technical reasons.

7.3.2.3.4 The most frequent use of the User-abort service will be to complete the termination of a dialogue following the successful execution of one of the transfer procedures or cancellation of coordination using the End service.

7.3.2.3.5 As another example of an operational reason, suppose that one ATSU becomes aware of conditions, such as equipment failure, that invalidate the assumptions underlying the use of AIDC to support transfer across one part of the boundary of its FIR. Then the ATSU's system management may wish to shut down the AIDC capability of the ATSU for that boundary immediately, without completing any current dialogues relevant to that boundary. An AIDC implementation would typically invoke the User-abort service for each such dialogue before shutting down the relevant capability, so that the peer implementation becomes aware of the dialogue termination quickly and its host system and users can take appropriate action. Without the use of User-abort, underlying connections will remain open until the expiration of an appropriate timer; timers are maintained by the AIDC-AE as well as in the ATN Transport service.

7.3.2.3.6 As an example of a technical reason, suppose that some set of dialogues is being supported by a single operating system process in a computer system of one ATSU, and that this process experiences a fault, causing it to be terminated. An AIDC implementation might save its state to enable such a process to be restarted; alternatively it may contain error handling functionality to invoke User-abort for each dialogue, with the same benefits described above.

7.3.2.3.7 As a result of limitations in the supporting upper-layer services, it is not possible to transmit any diagnostic information with the User-abort service. If there is important diagnostic information of an operational nature that needs to be brought to the attention of a controller in the peer ATSU, implementors may consider as an option using the Info-transfer service before the User-abort service; in this case it would be necessary to wait for the response (User-confirmation.indication) to the Info-transfer.request before issuing User-abort.request.

7.3.2.4 ***Provider-abort***

7.3.2.4.1 The Provider-abort service does not correspond to any message defined in Part VI.5 of [7]. It is delivered by the AIDC service to indicate an error in the service.

7.3.2.4.2 The Provider-abort service comprises the delivery of Provider-abort.indication to one or both AIDC-Users.

7.3.2.4.3 The parameter ProviderAbortReason supplied in the Provider-abort.indication distinguishes the various cases:

a) abort initiated by local or remote AIDC-AE, for example, as a result of the receipt of invalid data or the expiration of a timer within the AIDC-AE ('protocolerror', 'timerexpired', undefinederror');

b) abort initiated by local AIDC-AE, as a result of failing to establish an association ('rejectedtransient', 'rejectedpermanent');

c) abort initiated by presentation service or supporting layers ('providererror').

7.3.2.5 In case (a), it should be noted that the 'timerexpired' result is only provided to the local AIDC-User. The remote AIDC-User would receive 'protocolerror'.

7.3.2.6 In case (b), the error information that would have been available as the Result Source parameter of the A-ASSOCIATE.confirmation primitive is made available as a parameter of the Provider-abort.indication primitive.

7.3.2.7 ***Info-transfer***

7.3.2.7.1 The Info-transfer service implements the SurvGeneral, GeneralPoint, GeneralExecData, FreetextEmergency and FreetextGeneral messages defined in Part VI.5 of [7].

7.3.2.7.2 The Info-transfer service is unconfirmed: ie, the use of the service consists of the submission of Info-transfer.request by one AIDC-User and the corresponding delivery of Info-transfer.indication to the peer AIDC-User.

7.3.2.7.3 The Info-transfer service is used by one ATSU to transfer information to its peer ATSU about a flight.

7.3.2.7.4 The ATSU responds to the Info-transfer.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the originating ATSU's AIDC-AE.

7.3.2.7.5 The Info-transfer service with the parameter Information having the abstract syntax GeneralExecutiveData is used by an ATSU to transfer contact frequency and optionally executive control information (assigned heading, clearance, etc) for a flight. This corresponds to the GeneralExec message defined in Part VI.5 of [7].

7.3.2.7.6 The Info-transfer service with the parameter Information having the abstract syntax GeneralPoint is used by an ATSU to indicate a flight to a specified function (for instance, the supervisor) within the other ATSU, and to transfer flight plan information for the flight. This corresponds to the GeneralPoint message defined in Part VI.5 of [7].

7.3.2.7.7 The Info-transfer service with the parameter Information having the abstract syntax SurveillanceData is used by an ATSU to transfer track data (three-dimensional position and velocity) for a flight to its peer ATSU. This corresponds to the SurvGeneral message defined in Part VI.5 of [7].

7.3.2.7.8 The Info-transfer service with the parameter Information having the abstract syntax GeneralFreeText is used by an ATSU to transfer arbitrary text information relevant to a flight to a specified function (for instance, the supervisor) within the other ATSU. This corresponds to the FreetextGeneral message defined in Part VI.5 of [7].

7.3.2.7.9 The Info-transfer service with the parameter Information having the abstract syntax EmergencyFreeText is used by an ATSU to transfer arbitrary text information relevant to a flight in an emergency condition to a specified function (for instance, the supervisor) within the other ATSU. This corresponds to the FreetextEmergency message defined in Part VI.5 of [7].

7.3.2.7.10 The flight information that has to be transmitted with the Info-transfer service depends on the which syntax variant is used. The Info-transfer service also allows a wide variety of other flight information to be transferred; the elements that are actually sent will depend on the operational circumstances. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.2.7.11 In principle the Info-transfer service can be invoked at any time during a dialogue by either ATSU. The C-ATSU can also invoke the Info-transfer service before any dialogue has been initiated; in this case the Info-transfer service itself will initiate a dialogue. However, from an operational viewpoint, as defined in Part VI.5 of [7] (Table 5-1), the GeneralExec variant of the Info-transfer service is only supposed to be invoked after coordination conditions for the flight have been successfully agreed, and this would also typically be the case for the SurvGeneral, FreetextEmergency and FreetextGeneral; the AIDC-AE does not enforce such operational restrictions. The conditions under which each variant of the Info-transfer service are used will also have to be agreed between the ATSUs involved.

7.3.3 Notifying regime

7.3.3.1 *Notify*

7.3.3.1.1 The Notify service implements the Notify message defined in Part VI.5 of [7].

7.3.3.1.2 The Notify service is unconfirmed: ie, the use of the service consists of the submission of Notify.request by one AIDC-User and the corresponding delivery of Notify.indication to the peer AIDC-User.

7.3.3.1.3 The Notify service is used by the C-ATSU to inform the D-ATSU of a flight that will be the subject of coordination and transfer and to provide updated information about the flight.

7.3.3.1.4 The D-ATSU responds to the Notify.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU’s AIDC-AE.

7.3.3.1.5 The flight information that has to be transmitted with the Notify service comprises the aircraft identification and the position, time and flight level at which it is estimated to cross the boundary. The Notify service also allows a wide variety of other flight information to be transferred; the elements that are actually sent will depend on the operational circumstances. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.3.1.6 The Notify service will typically be invoked well before the flight reaches the boundary. The amount of notice will depend on the local conditions; this will also have to be agreed between the ATSUs involved.

7.3.3.1.7 A D-ATSU may receive a Notify.Indication for a flight for which it holds no flight plan. If sufficient flight plan information is included, the receiving system may be able to create a suitable flight plan. Otherwise, the information may be rejected using the User-confirmation service, indicating the component type as 'FlightID' and the error code as 'invalidAircraftFlightID'. The C-ATSU may in turn invoke the Notify service again, sending sufficient additional flight information for the D-ATSU to construct a flight plan. Alternatively, other mechanisms, as described in 7.2.12.1, may be deployed, or ultimately manual intervention may be needed.

7.3.4 Coordinating regime

7.3.4.1 *Coordinate start*

7.3.4.1.1 The Coordinate-start service implements the CoordinateInitial and CoordinateUpdate messages defined in Part VI.5 of [7].

7.3.4.1.2 The Coordinate-start service is unconfirmed: ie, the use of the service consists of the submission of Coordinate-start.request by one AIDC-User and the corresponding delivery of Coordinate-start.indication to the peer AIDC-User.

7.3.4.1.3 The Coordinate-start service with the parameter Coordinate Start Information having the abstract syntax CoordinateInitial is used by the C-ATSU to initiate a coordination dialogue with the D-ATSU for a flight that will be the subject of coordination and transfer. At the same time the C-ATSU provides updated information about the flight. This corresponds to the CoordinateInitial message defined in Part VI.5 of [7].

7.3.4.1.4 The Coordinate-start service with the parameter Coordinate Start Information having the abstract syntax CoordinateUpdate is used by the C-ATSU to reinitiate a previously concluded coordination dialogue with the D-ATSU for a flight that will be the subject of coordination and transfer. At the same time the C-ATSU provides updated information about the flight. This corresponds to the CoordinateUpdate message defined in Part VI.5 of [7].

7.3.4.1.5 The Coordinate-start service with the parameter Coordinate Start Information having the abstract syntax CoordinateUpdate may also be used by an R-ATSU to initiate a backward coordination dialogue with the C-ATSU for a flight that has been transferred but has not yet left a defined area of common interest on either side of the FIR boundary.

7.3.4.1.6 The D-ATSU responds to the Coordinate-start.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU's AIDC-AE.

7.3.4.1.7 The flight information that has to be transmitted with the Coordinate-start service comprises the aircraft identification and the position, time and flight level at which it is estimated to cross the boundary. The Coordinate-start service also allows a wide variety of other flight information to be transferred; the elements that are actually sent will depend on the operational circumstances. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.4.1.8 The Coordinate-start service will typically be invoked in good time to ensure that the result of the coordination dialogue is available before the flight reaches the boundary, but late enough that the result will be accurate. The time before the boundary at which coordination is initiated will depend on the local conditions; this will also have to be agreed between the ATSUs involved.

7.3.4.1.9 If the Notify service has not been used, the use of the Coordinate-start service with the CoordinateInitial syntax also fulfils the role of the Notify service. This use may be required to implement pre-departure coordination where the flight is departing from an airport close to the boundary and an agreement is in place that specifies a minimum time to the boundary after which Notify may not be used. The issues, particularly concerning the processing of a new flight, described in 7.3.3.1 apply equally to the Coordinate-start service in this case.

7.3.4.1.10 The boundary estimate provided in the Coordinate-start.indication constitutes the C-ATSU's proposal for the coordination conditions for the flight. The D-ATSU's User-confirmation response does not take account of whether the proposed coordination conditions are acceptable from an operational viewpoint. Once the D-ATSU has analysed the proposed coordination conditions, it responds in one of three ways:

a) if the proposed coordination conditions are accepted, the D-ATSU uses the Coordinate-end service;

b) if the proposed coordination conditions are not accepted, the D-ATSU uses the Coordinate-negotiate service to propose alternative conditions;

c) while the acceptability of the proposed coordination conditions cannot yet be determined, the D-ATSU may use the Coordinate-standby service (repeatedly, if necessary) to extend the C-ATSU's waiting limit.

7.3.4.1.11 When two ATSUs agree to use AIDC in support of coordination and transfer across their mutual boundary, they may establish a set of standard coordination conditions, related to the route structure at the boundary. A coordination proposal that matches one of the standard conditions may be accepted automatically, while other proposals require controller intervention (or, in the future, at least more lengthy automated processing) to be accepted. In this situation, the Coordinate-end response can be generated immediately following the automatic acceptance of the proposed conditions; the Coordinate-standby response is generated while a proposal is being analysed; the Coordinate-negotiate response is generated if the controller rejects the proposal (for instance, it may instead propose coordination conditions matching one of the standard coordination conditions).

7.3.4.2 *Coordinate-end*

7.3.4.2.1 The Coordinate-end service implements the CoordinateAccept and CoordinateReject messages defined in Part VI.5 of [7].

7.3.4.2.2 The Coordinate-end service is unconfirmed: ie, the use of the service consists of the submission of Coordinate-end.request by one AIDC-User and the corresponding delivery of Coordinate-end.indication to the peer AIDC-User.

7.3.4.2.3 The Coordinate-end service is used by an ATSU in a coordination dialogue to end the coordination phase, whether successfully or unsuccessfully.

7.3.4.2.4 The ATSU responds to the Coordinate-end.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the peer ATSU's AIDC-AE.

7.3.4.2.5 The Coordinate-end service may also be used by an R-ATSU to conclude a backward coordination dialogue with the C-ATSU for a flight that has been transferred but has not yet left a defined area of common interest on either side of the FIR boundary.

7.3.4.2.6 When the Result parameter provided with the Coordinate-end.indication has the abstract value 'accepted', the Coordinate End Information parameter has the syntax CoordinateAccept and the submitting ATSU is signalling its acceptance of the coordination conditions most recently proposed by the receiving ATSU using either Coordinate-start or Coordinate-negotiate. The coordination dialogue is concluded; subsequently the coordination dialogue may be reopened using Coordinate-start with the CoordinateUpdate syntax, or the transfer regime may be initiated. This corresponds to the CoordinateAccept message defined in Part VI.5 of [7].

7.3.4.2.7 When the Result parameter provided with the Coordinate-end.indication has the abstract value 'rejected', the Coordinate End Information parameter has the syntax CoordinateReject and the submitting ATSU is signalling its rejection of the coordination conditions most recently proposed by the receiving ATSU using either Coordinate-start or Coordinate-negotiate. Any previous coordination conditions remain as agreed. This corresponds to the CoordinateReject message defined in Part VI.5 of [7].

7.3.4.3 *Coordinate-negotiate*

7.3.4.3.1 The Coordinate-negotiate service implements the CoordinateNegotiate message defined in Part VI.5 of [7].

7.3.4.3.2 The Coordinate-negotiate service is unconfirmed: ie, the use of the service consists of the submission of Coordinate-negotiate.request by one AIDC-User and the corresponding delivery of Coordinate-negotiate.indication to the peer AIDC-User.

7.3.4.3.3 The Coordinate-negotiate service is used by an ATSU in a coordination dialogue to negotiate the coordination conditions for a flight that will be the subject of coordination and transfer. At the same time the submitting ATSU provides updated information about the flight.

7.3.4.3.4 The Coordinate-negotiate service may also be used by an ATSU during a backward coordination dialogue to negotiate a proposed change to a transferred flight's profile while it is still in a defined area of common interest on either side of the boundary.

7.3.4.3.5 The ATSU responds to the Coordinate-negotiate.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the peer ATSU's AIDC-AE.

7.3.4.3.6 The flight information that has to be transmitted with the Coordinate-negotiate service comprises the aircraft identification and the position, time and flight level at which it is estimated to cross the boundary. The Coordinate-negotiate service also allows a wide variety of other flight information to be transferred; the elements that are actually sent will depend on the operational circumstances. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.4.3.7 The boundary estimate provided in the Coordinate-negotiate.indication constitutes the ATSU's revised proposal for the coordination conditions for the flight. The processing of the proposed coordination conditions by the receiving ATSU is the same as that described for the D-ATSU in 7.3.4.1.

7.3.4.3.8 The AIDC service places no limit on the number of times that each AIDC-User may respond with Coordinate-negotiate to proposals made in the other ATSU's Coordinate-start or Coordinate-negotiate before one or other uses Coordinate-end. There is always a practical limit on the time that can be spent in the coordination dialogue, considering the normal progress of the flight. If the dialogue is not concluded in a timely fashion it is open to either party to terminate the dialogue using the User-abort service and conclude the coordination by other means. In the case of backward coordination, the flight may progress outside the area of common interest, so removing the need for backward coordination.

7.3.4.4 *Coordinate-standby*

7.3.4.4.1 The Coordinate-standby service implements the CoordinateStandby message defined in Part VI.5 of [7].

7.3.4.4.2 The Coordinate-standby service is unconfirmed: ie, the use of the service consists of the submission of Coordinate-standby.request by one AIDC-User and the corresponding delivery of Coordinate-standby.indication to the peer AIDC-User.

7.3.4.4.3 The Coordinate-standby service is used by an ATSU in receipt of a coordination proposal to inform its peer ATSU to extend its waiting time.

7.3.4.4.4 The Coordinate-standby service may also be used during a backward coordination dialogue by an ATSU in receipt of a coordination proposal to inform its peer ATSU to extend its waiting time.

7.3.4.4.5 The ATSU responds to the Coordinate-standby.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU's AIDC-AE.

7.3.4.4.6 The Coordinate-standby service will typically be invoked when a coordination proposal has been received that does not match any of the pre-arranged standard coordination conditions agreed between the two ATSUs and consequently requires controller intervention. In future systems that support greater integration of flight planning across multiple FIRs, it could be used to extend the waiting time while more extensive automated analysis of the proposal is undertaken.

7.3.4.4.7 The AIDC service places no limit on the number of times that the AIDC-User may respond with Coordinate-standby before using Coordinate-end or Coordinate-negotiate. There is always a practical limit on the time that can be spent in the coordination dialogue, considering the normal progress of the flight. If the dialogue is not concluded in a timely fashion it is open to either party to terminate the dialogue using the User-abort service and conclude the coordination by other means. In the case of backward coordination, the flight may progress outside the area of common interest, so removing the need for backward coordination.

7.3.5 Transferring regime

7.3.5.1 *Transfer options*

7.3.5.1.1 AIDC supports two approaches to the transfer of control and communications authority for a flight.

7.3.5.1.2 The first approach is suitable only for transfers across an FIR boundary that has full surveillance using radar or ADS. In this case the position of the aircraft in relation to the boundary is known to some adequate precision and transfer of communications authority can imply transfer of full control and communications authority (as typically documented in a Letter of Agreement between the two ATSUs). This approach uses the Transfer-initiate service, either the Transfer-communication service or the Transfer-communication-assume service, optionally the Transfer-request or Transfer-conditions-proposal services, and, if mutually agreed, the Transfer-conditions-accept service.

7.3.5.1.3 The second approach is suitable for transfers across an FIR boundary that does not have full surveillance. In this case an area of common interest is defined on either side of the boundary such that changes to a flight's profile under control of one ATSU have to be coordinated with the other ATSU. After the transfer of control to the R-ATSU it does not assume full control and communications authority until the flight leaves the boundary region. This approach uses the Transfer-control service.

7.3.5.1.4 Two ATSUs using AIDC to support transfer of flights across a FIR boundary have to agree on which of the above approaches are to be used.

7.3.5.2 *Transfer-initiate*

7.3.5.2.1 The Transfer-initiate service implements the TransferInitiate message defined in Part VI.5 of [7].

7.3.5.2.2 The Transfer-initiate service is unconfirmed: ie, the use of the service consists of the submission of Transfer-initiate.request by one AIDC-User and the corresponding delivery of Transfer-initiate.indication to the peer AIDC-User.

7.3.5.2.3 The Transfer-initiate service is used by the C-ATSU to initiate the transfer phase of a dialogue with the R-ATSU for a flight that is to be transferred. At the same time the C-ATSU provides executive control information about the flight.

7.3.5.2.4 The R-ATSU responds to the Transfer-initiate.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU's AIDC-AE.

7.3.5.2.5 The Transfer-initiate service allows various types of flight information (specifically, track and executive control data) to be transferred; the elements that are actually sent will depend on the operational circumstances. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.5.2.6 The Transfer-initiate service will typically be invoked, once the result of the coordination dialogue is available, as the flight reaches the boundary. The service may be initiated by the C-ATSU on recognising that the flight is approaching the boundary, or the C-ATSU may be responding to a Transfer-request primitive generated by the R-ATSU on recognising the need to assume control of the flight. The time before the boundary at which transfer is initiated will depend on the local conditions; this will also have to be agreed between the ATSUs involved.

7.3.5.3 *Transfer-request*

7.3.5.3.1 The Transfer-request service implements the TransferRequest message defined in Part VI.5 of [7].

7.3.5.3.2 The Transfer-request service is unconfirmed: ie, the use of the service consists of the submission of Transfer-request.request by one AIDC-User and the corresponding delivery of Transfer-request.indication to the peer AIDC-User.

7.3.5.3.3 The Transfer-request service is used by the R-ATSU to request the C-ATSU to initiate the transfer phase of a dialogue for a flight that is to be transferred. The request may include the frequency on which the R-ATSU intends to communicate with the aircraft.

7.3.5.3.4 The C-ATSU responds to the Transfer-request.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the R-ATSU's AIDC-AE.

7.3.5.3.5 The Transfer-request service will typically be invoked, once the result of the coordination dialogue is available, as the flight is approaching the boundary and when the R-ATSU recognises the need to assume control and communications authority for the flight.

7.3.5.4 *Transfer-conditions-proposal*

7.3.5.4.1 The Transfer-conditions-proposal service implements the TransferConditionsProposal message defined in Part VI.5 of [7].

7.3.5.4.2 The Transfer-conditions-proposal service is unconfirmed: ie, the use of the service consists of the submission of Transfer-conditions-proposal.request by one AIDC-User and the corresponding delivery of Transfer-conditions-proposal.indication to the peer AIDC-User.

7.3.5.4.3 The Transfer-conditions-proposal service is used by the C-ATSU to propose the conditions under which a flight will be transferred to the R-ATSU.

7.3.5.4.4 The R-ATSU responds to the Transfer-conditions-proposal.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU's AIDC-AE.

7.3.5.4.5 The Transfer-conditions-proposal service allows the C-ATSU to propose executive control information (that is, a clearance) for the flight to be accepted by the R-ATSU, and to transfer other flight information. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.5.4.6 The Transfer-conditions-proposal service will typically be invoked following acknowledgement of the Transfer-initiate service and before the flight reaches the boundary.

7.3.5.4.7 Following the acknowledgement of the Transfer-conditions-proposal service, several actions are possible:

a) the R-ATSU can accept the proposal using the Transfer-conditions-accept service;

b) the C-ATSU can relinquish control and communications authority for the flight using the Transfer-communication service;

c) the R-ATSU can assume control and communications authority for the flight using the Transfer-communication-assume service.

7.3.5.4.8 The choice between using the Transfer-conditions-accept service and using either the Transfer-communication service or the Transfer-communication-assume service is fixed and needs to be agreed between any two ATSUs that use AIDC to support transfer of flights.

7.3.5.5 *Transfer-conditions-accept*

7.3.5.5.1 The Transfer-conditions-accept service implements the TransferConditionsAccept message defined in Part VI.5 of [7].

7.3.5.5.2 The Transfer-conditions-accept service is unconfirmed: ie, the use of the service consists of the submission of Transfer-conditions-accept.request by one AIDC-User and the corresponding delivery of Transfer-conditions-accept.indication to the peer AIDC-User.

7.3.5.5.3 The Transfer-conditions-accept service is used by the R-ATSU to accept the C-ATSU's proposal of conditions for the transfer of a flight.

7.3.5.5.4 As indicated in SARPs [2] 3.2.9.1.1d) and 3.2.1.6.4, the use of the Transfer-conditions-accept service rather than either the Transfer-communication service or the Transfer-communication-assume service in response to the Transfer-conditions-proposal service is fixed and needs to be agreed between any two ATSUs that use AIDC to support transfer of flights.

7.3.5.5.5 The C-ATSU responds to the Transfer-conditions-accept.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the R-ATSU's AIDC-AE.

7.3.5.5.6 The Transfer-conditions-proposal service allows the R-ATSU to inform the C-ATSU of a frequency on which it can be contacted by the flight, and to transfer other flight information. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.5.5.7 The Transfer-conditions-accept service will typically be invoked to accept the proposed transfer conditions before proceeding either to receive or to assume communications authority for the flight.

7.3.5.6 *Transfer-communication*

7.3.5.6.1 The Transfer-communication service implements the TransferComm message defined in Part VI.5 of [7].

7.3.5.6.2 The Transfer-communication service is unconfirmed: ie, the use of the service consists of the submission of Transfer-communication.request by one AIDC-User and the corresponding delivery of Transfer-communication.indication to the peer AIDC-User.

7.3.5.6.3 The Transfer-communication service is used by the C-ATSU to relinquish communications authority for a flight to the R-ATSU.

7.3.5.6.4 The R-ATSU responds to the Transfer-communication.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU's AIDC-AE.

7.3.5.6.5 The Transfer-communication service allows the C-ATSU to transfer executive control information for the flight as well as the conditions under which the flight is released to the R-ATSU. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.5.6.6 The Transfer-communication service will typically be invoked during the transfer dialogue before the flight reaches the boundary when the C-ATSU recognises that it has no further need to maintain control over the flight. As indicated above, the C-ATSU can place conditions on the release of the flight to avoid the possibility that the R-ATSU modifies the flight's profile so as to create a conflict with other traffic under the control of the C-ATSU.

7.3.5.7 *Transfer-communication-assume*

7.3.5.7.1 The Transfer-communication-assume service implements the TransferCommAssume message defined in Part VI.5 of [7].

7.3.5.7.2 The Transfer-communication-assume service is unconfirmed: ie, the use of the service consists of the submission of Transfer-communication-assume.request by one AIDC-User and the corresponding delivery of Transfer-communication-assume.indication to the peer AIDC-User.

7.3.5.7.3 The Transfer-communication-assume service is used by the R-ATSU to assume communications authority for a flight from the C-ATSU.

7.3.5.7.4 The C-ATSU responds to the Transfer-communication-assume.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the R-ATSU's AIDC-AE.

7.3.5.7.5 The Transfer-communication-assume service will typically be invoked during the transfer dialogue before the flight reaches the boundary when the R-ATSU recognises that it has a need to assume control over the flight.

7.3.5.8 *Transfer-control*

7.3.5.8.1 The Transfer-control service implements the TransferControl, TransferControlAssume and TransferControlReject messages defined in Part VI.5 of [7].

7.3.5.8.2 The Transfer-control service is confirmed: ie, the use of the service consists of the submission of Transfer-control.request by one AIDC-User, the corresponding delivery of Transfer-control.indication to the peer AIDC-User, submission of

Transfer-control.response by that AIDC-User, and delivery of Transfer-control.confirmation to the originating AIDC-User.

7.3.5.8.3 The Transfer-control service is used by the C-ATSU to transfer control of a flight to the R-ATSU.

7.3.5.8.4 The Transfer-control.request and Transfer-control.indication primitives correspond to the TransferControl message defined in Part VI.5 of [7].

7.3.5.8.5 The R-ATSU responds to the Transfer-control.indication using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the C-ATSU's AIDC-AE.

7.3.5.8.6 When the Result parameter provided with the Transfer-control.confirmation has the abstract value 'accepted', the Transfer Control Information parameter has the syntax TransferAccept and the R-ATSU is signalling its acceptance of the transfer of control. This corresponds to the TransferAccept message defined in Part VI.5 of [7].

7.3.5.8.7 When the Result parameter provided with the Transfer-control.confirmation has the abstract value 'rejected', the Transfer Control Information parameter has the syntax TransferReject and the R-ATSU is signalling its rejection of the transfer of control. This corresponds to the TransferReject message defined in Part VI.5 of [7].

7.3.5.8.8 The Transfer-control.response and Transfer-control.confirmation primitives correspond to the TransferControlAssume message defined in Part VI.5 of [7].

7.3.5.8.9 The C-ATSU responds to the Transfer-control.confirmation using the User-confirmation service, depending on its success in accepting the information provided. This response has to be generated in accordance with 7.4.7.3 to avoid exceeding the time limit set by the R-ATSU's AIDC-AE.

7.3.5.8.10 The Transfer-control service allows the C-ATSU to transfer executive control information for the flight. Rules determining the components to be sent should in any case have been agreed on a regional or bilateral basis.

7.3.5.8.11 The Transfer-control service will typically be invoked during the transfer dialogue before the flight reaches the boundary when the C-ATSU recognises that it has no further need to maintain control over the flight.

7.3.6 AIDC Service Primitive Sequencing

7.3.6.1 Table 7.3-1 provides information on the sequencing of the AIDC service primitives. This table does not show the common sequencing of the User-confirmation service primitives.

Table 7.3-1. AIDC service primitives sequencing

AIDC service primitive	Source/ Destination	Valid operational response
Notify Service Primitive	C-ATSU / D-ATSU	None required
Coordinate-start Service Primitive, with Coordinate Start Information parameter of type *CoordinateInitial*	C-ATSU / D-ATSU	C Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateAccept*; or C Coordinate-negotiate Service Primitive *Note.— Operational response may be deferred using the Coordinate-standby service*
Coordinate-negotiate Service Primitive	ATSU1 / ATSU2	C Coordinate-negotiate Service Primitive; or C Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateAccept*; or C Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateReject* *Note.— Operational response may be deferred using the Coordinate-standby service*
Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateAccept*	ATSU1 / ATSU2	None required
End Service Primitive	C-ATSU / D-ATSU	None required
Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateReject*	ATSU1 / ATSU2	None required

AIDC service primitive	Source/ Destination	Valid operational response
Coordinate-start Service Primitive, with Coordinate Start Information parameter of type *CoordinateUpdate*	ATSU1 / ATSU2	C Coordinate-negotiate Service Primitive; or C Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateAccept*; or C Coordinate-end Service Primitive, with Coordinate End Information parameter of type *CoordinateReject* *Note.— Operational response may be deferred using the Coordinate-standby service*
Coordinate-standby Service Primitive	ATSU1 / ATSU2	None required (ATSU1 must follow with the previously expected operational response)
Transfer-initiate Service Primitive	C-ATSU / R-ATSU	C None required (C-ATSU would typically follow with Transfer-conditions-proposal.request service primitive)
Transfer-conditions-proposal Service Primitive	C-ATSU / R-ATSU	C None required; or C Transfer-conditions-accept Service Primitive
Transfer-conditions-accept Service Primitive	R-ATSU / C-ATSU	C None required (R-ATSU would typically follow with Transfer-communications-assume.request); or C Transfer-communications Service
Transfer-request Service Primitive	R-ATSU / C-ATSU	None required (C-ATSU would typically repond with Transfer-initiate Service Primitive)
Transfer-control Service Request/ Indication Primitive, with Transfer Control Information parameter of type *TransferControl*	C-ATSU / R-ATSU	C Transfer-control Service Response Primitive, with Transfer Control Information parameter of type *TransferControlAssume* and Result parameter = *Accept*; or C Transfer-control Service Response Primitive, with Transfer Control Information parameter of type *TransferControlReject* and Result parameter = *Reject*

AIDC service primitive	**Source/ Destination**	**Valid operational response**
Transfer-control Service Response/ Confirmation Primitive, with Transfer Control Information parameter of type *TransferControl*	R-ATSU / C-ATSU	C Transfer-control Service Response Primitive, with Transfer Control Information parameter of type *TransferControlAssume* and Result parameter = *Accept*; or C Transfer-control Service Response Service Primitive, with Transfer Control Information parameter of type *TransferControlReject* and Result parameter = *Reject*;
Transfer-control Service Response/ Confirmation Primitive, with Transfer Control Information parameter of type *TransferControlAssume* and Result parameter = *Accept*	R-ATSU / C-ATSU	None required (R-ATSU may issue a Coordinate-start.request primitive using the CoordinateUpdate syntax; otherwise either ATSU may issue User-abort once aircraft leaves the area of common interest)
Transfer-control Service Response/ Confirmation Primitive, with Transfer Control Information parameter of type *TransferControlReject* and Result parameter = *Reject*	R-ATSU / C-ATSU	None required (C-ATSU may issue a Coordinate-start.request primitive using the CoordinateUpdate syntax to negotiate new coordination conditions; otherwise either ATSU may issue User-abort and revert to manual procedures)
Transfer-communication Service Primitive	C-ATSU / R-ATSU	C None required; or C Transfer-communication-assume Service Primitive
Transfer-communication-assume Service Primitive	R-ATSU / C-ATSU	C None required (either ATSU may issue User-abort following transfer of control)
Info-transfer Service Primitive, with Information parameter of type *GeneralExecutiveData* or *GeneralPoint*, *SurveillanceData* or *GeneralFreeText* or *EmergencyFreeText*	ATSU1 / ATSU2	None required

7.4 Commentary on the AIDC SARPs

7.4.1 Introduction

7.4.1.1 The text in this section provides additional guidance on each section of the SARPs [2] for AIDC (3.2).

7.4.2 SARPs 3.2.2 (Introduction)

7.4.2.1 Figure 7.4-1 shows the elements of the SARPs and their relationship to the functional model shown in the SARPs.

7.4.3 SARPs 3.2.2 (General Requirements)

7.4.3.1 *Error Handling*

7.4.3.1.1 The use of AIDC to support coordination and transfer of flights across a FIR boundary will be based on a regional or bilateral agreement that selects a subset of the AIDC service and protocol in accordance with the operational needs of the ATSUs.

7.4.3.1.2 AIDC provides the User-abort service that enables either AIDC-User to abort any dialogue in the event of either system failure or operational errors, such as the transmission of information outside the bounds of the governing agreement. However, excessive use of this facility will nullify the benefits of automation that may result from the use of AIDC.

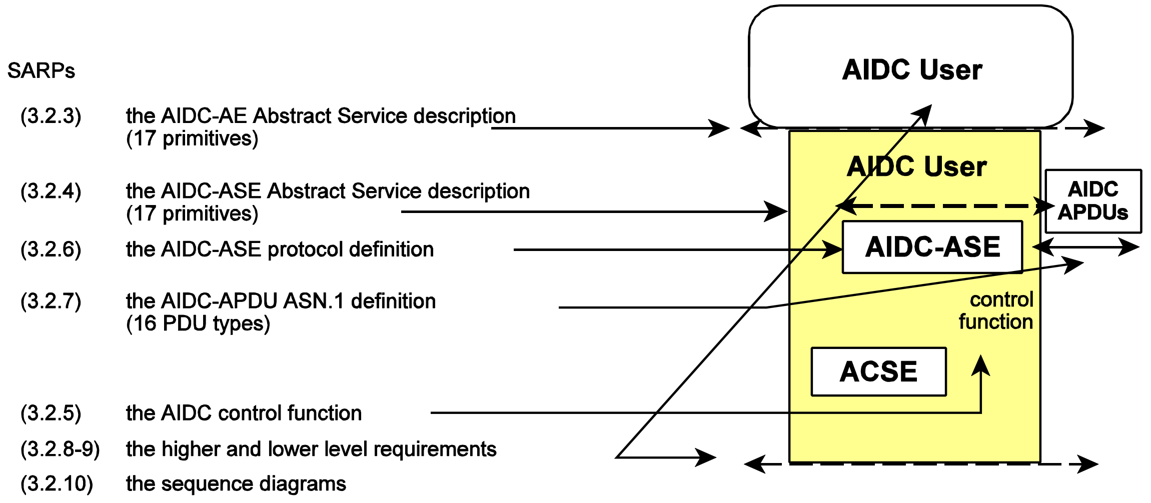

SARPs
(3.2.3) the AIDC-AE Abstract Service description (17 primitives)
(3.2.4) the AIDC-ASE Abstract Service description (17 primitives)
(3.2.6) the AIDC-ASE protocol definition
(3.2.7) the AIDC-APDU ASN.1 definition (16 PDU types)
(3.2.5) the AIDC control function
(3.2.8-9) the higher and lower level requirements
(3.2.10) the sequence diagrams
AIDC User
AIDC User
AIDC APDUs
AIDC-ASE
control function
ACSE

Figure 7.4-1. Contents of the AIDC SARPs

7.4.3.1.3 An implementation of the AIDC-AE should be capable of handling any optional parameter unless the use of the implementation is such that the parameter is certain never to be transmitted.

7.4.3.1.4 Implementors of the AIDC-User components should pay attention wherever possible to ignoring unexpected data fields rather than rejecting the communication or aborting the dialogue.

7.4.3.2 *Use of ULCS in AIDC*

7.4.3.2.1 The AIDC application uses the Upper Layer Communications Service specified in SARPs 4. Unlike the air-ground applications, it is not specified in terms of the Dialogue Service defined in SARPs 4, but it uses the same protocol elements in a way that should allow a Dialogue Service implementation to be used for AIDC as well. This is further described in 7.4.6.

7.4.4 SARPs 3.2.3 (The AIDC-AE Abstract Service)

7.4.4.1 *Specification of the abstract service*

7.4.4.1.1 The AIDC-AE Abstract Service is defined using the conventions for the definitions of OSI services, as described in 7.2.5.

7.4.4.1.2 The parameters supported by each AIDC service are presented in tables of the following form.

Table 7.4-1. Example of parameter specification (Notify)

Parameter Name	Req	Ind
Called ICAO Facility Designation	M	
Calling ICAO Facility Designation		M
Notification Information	M	M(=)
Message Number	M	M(=)

7.4.4.1.3 The parameter is defined by its name, shown in the first column.

7.4.4.1.4 The subsequent columns specify which parameters are used in the various primitives of the service. For an unconfirmed service, there are only request (usually) and indication primitives. For a confirmed service there will be columns for response and confirmation primitives.

7.4.4.1.5 In these columns, the letter ‘M’ means that it is mandatory to provide the parameter. The letter ‘U’ means that the parameter is optional. The letter ‘C’ in an indication or confirmation column means that the parameter is mandatory for that primitive if it was provided in the corresponding request or response primitive. Finally, the symbol'=' appended to M or C means that the parameter must have the same value as that provided in the corresponding preceding primitive invocation; thus, in the example above, the Notification Information and Message Number in the Notify.indication primitive must have the same values as those in the Notify.request primitive.

7.4.4.1.6 Each parameter is further specified in the accompanying text by specifying the abstract syntax of the parameter. The abstract syntax is defined by reference to the ASN.1 module AIDCMessageSetVersion1 in SARPs 3.2.7.1 (the use of ASN.1 is further discussed in 7.4.8). This is not meant to imply that data should be passed over some internal boundary between the AIDC-User and the AIDC-AE using one of the defined encoding rules for ASN.1; typically the data elements modelled by the ASN.1 specification of a service parameter will be represented at this level as types in whatever programming language is chosen for the implementation. However, the ASN.1 specification provides a standard definition of the parameter syntax. In addition, the specification provides the essential link between the service parameters and the data transmitted in the AIDC protocol, whose syntax is defined in the same ASN.1 module.

7.4.5 **SARPs 3.2.4 (The AIDC-ASE Abstract Service)**

7.4.5.1 ***Function***

7.4.5.1.1 The AIDC-ASE is the functional module that models the protocol associated with the AIDC application.

7.4.5.1.2 As modelled, the AIDC-ASE is only active when there is AIDC data to be transferred between the AIDC-AE peers.

7.4.5.1.3 The AIDC-ASE is not involved with the establishment of the association between AIDC-ASE peers.

7.4.5.2 ***AIDC-ASE Services***

7.4.5.2.1 The abstract service boundary defined for the AIDC-ASE has two components:

a) the upper service boundary, defined in SARPs 3.2.4.2, corresponds closely to the AIDC-AE abstract service defined in SARPs 3.2.3;

b) the lower service boundary, defined in SARPs 3.2.4.3, is an abstraction of the supporting communications service required by the ASE.

7.4.5.2.2 The AIDC-ASE, as specified, may also be used in the specification of other applications, provided that the corresponding control function is able to coordinate other ASEs to meet the AIDC-ASE service requirements.

7.4.6 SARPs 3.2.5 (The AIDC Control Function)

7.4.6.1 *Relation to ULCS and Correspondence to use of Dialogue Service*

7.4.6.1.1 Figure 7.4-2 shows the application-entity structure specified in the ULCS SARPs as used in the air-ground applications. These applications have been defined in terms of an abstract Dialogue Service provided using ACSE and the Presentation Service.

7.4.6.1.2 In this case the control function is effectively split into two parts by the Dialogue Service boundary:

a) an upper part, which we will call the ATN-App-AE control function, that maps between the ATN-application-user service primitives and the ATN-App-ASE primitives, and between the ATN-App-ASE and the Dialogue Service;

b) a lower part, which we will call the ULA control function, that implements the Dialogue Service by mapping between Dialogue Service primitives and those of ACSE and the Presentation Service.

7.4.6.1.3 In the applications so far defined, the ATN-App-AE control function is trivial because the ATN-application-user service is effectively identical to the upper service boundary of the

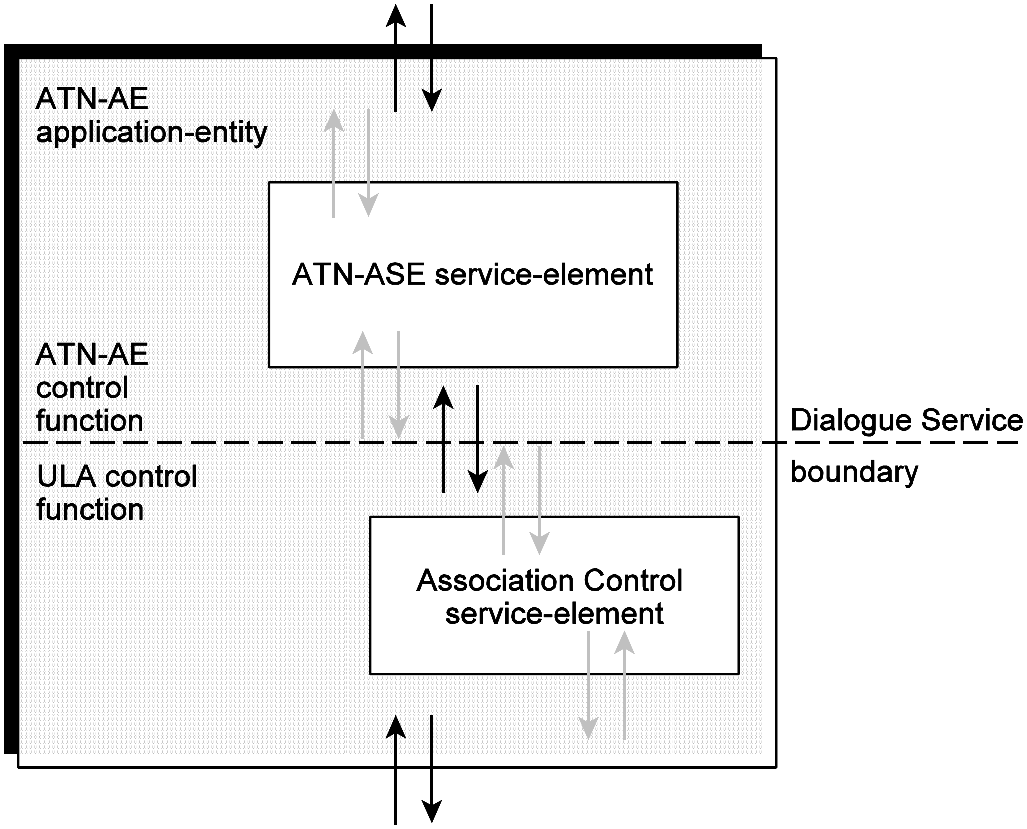


Figure 7.4-2. Structure of an ATN Application-entity

ATN-App-ASE, and the lower service boundary of the ATN-App-ASE is identical to the Dialogue Service boundary.

7.4.6.1.4 The ULA control function is very similar to the AIDC control function. The major difference concerns the specification of the implicit start — that is, when the application-user submits a data transfer service primitive that requires the creation of a dialogue. In AIDC, such a primitive (for example, Notify.request) is intercepted by the CF and mapped into an A-ASSOCIATE invocation to ACSE followed (if the association was successfully established) by data transfer using P-DATA. Using the Dialogue Service, the primitive would map to D-Start.request, which in turn is mapped into the A-ASSOCIATE and P-DATA invocations as described for AIDC. Thus the handling of the implicit start is done in the same way in the two cases, but above the AIDC-ASE by the AIDC CF and below the ATN-App-ASE (in the Dialogue Service) by the ULA CF.

7.4.6.1.5 Apart from this difference, the AIDC CF can be considered as a subset of the ULA CF. It should be possible to implement the AIDC CF and the CF based on the Dialogue Service using common code.

7.4.6.1.6 Like the ULCS, this version of the AIDC CF does not transmit association establishment ACSE PDUs (AARQ, AARE) using P-DATA, since they are small enough to be sent via P-CONNECT User Data. For the use of significant A-ASSOCIATE parameters, such as authentication-value, this mapping might be necessary. Both the AIDC CF and the ULCS perform a mapping of the A-ABORT service on to P-DATA in order to transmit diagnostics, since the FastByte P-U-ABORT does not support User Data.

7.4.6.2 *Function*

7.4.6.2.1 The CF modelled in the AIDC SARPs forms the basic framework which supports the other service elements modelled as part of the AIDC-AE.

7.4.6.2.2 The CF is responsible for the following:

a) mapping primitives submitted to the CF by the AIDC-User to the primitives of the appropriate ASEs with the AIDC-AE;

b) establish the association between AIDC-AE peers, implicitly, at the start of a new dialogue;

c) mapping primitives submitted by the ASEs within the AIDC-AE to other ASEs within the AIDC-AE and primitives at the ATN Service Provider interface;

d) mapping of primitives submitted to the CF from the ATN Service Provider interface to the ASEs within the AIDC-AE.

7.4.6.3 *Implementation*

7.4.6.3.1 The CF is not dissimilar to the main part of a software program, if the other elements of the AIDC-AE are viewed as software modules.

7.4.6.3.2 In order to facilitate the upgrading of the AIDC application from one version to another, where protocol is added, the CF should be implemented as a module.

7.4.6.4 ***State Machine Specification***

7.4.6.4.1 The CF specification is set out in the form of a finite state machine: that is, it is modelled as comprising a state which may have one of a finite number of values together with a finite set of events that may occur and a set of actions (including changing the state) that must be performed in response to each event.

7.4.6.4.2 In the AIDC CF, the state is structured to make the specification easier to express, by reducing the number of separate states. The state comprises:

a) the main state variable, taking one of the four values IDLE, ASSOCIATION PENDING, DATA TRANSFER, and RELEASE PENDING (these are also identified as STA0, STA1, STA2 and STA3 respectively);

b) a set of auxiliary state variables that define subsets of the main state:

1) nfy-assc, used to indicate that a pending association was initiated as a result of a Notify primitive;

2) crd-assc, used to indicate that a pending association was initiated as a result of a Coordinate-start primitive;

3) inf-assc, used to indicate that a pending association was initiated as a result of a Info-transfer primitive.

7.4.6.4.3 Although these variables are defined as boolean, they are not independent. At most one may be set at any time.

7.4.6.4.4 For each value of the main state and for each possible event, the SARPs provides a specification of the actions to be performed. Where the action depends on the subsets of the state, different actions are specified conditional on the auxiliary state variables.

7.4.6.4.5 The possible events depend on the current value of the state. When no action for an event is specified in the SARPs for some value of the state, the event is either logically impossible or not allowed by the protocol. In either case, an implementation of the AIDC-ASE is required to treat the occurrence of an event for which no action is specified in the SARPs as an exception according to the procedures in SARPs 3.2.6.2. Exception handling is discussed further in 7.4.7.2.

7.4.6.4.6 The AIDC CF state machine is summarised in the table presented in SARPs 3.2.5.4. The table should be a condensed version of the requirements stated in the preceding text and in case of a conflict the text specification takes precedence.

7.4.6.4.7 In the state table, each cell contains a specification of the actions to be performed for the given event and value of the main state variable. When different actions have to be performed depending on the auxiliary state variables, the cell contents comprises a series of specifications each introduced by a conditional expressions, using the keyword 'if', and followed by the actions for that set of conditions, culminating in a default set of actions to be performed if none of the preceding conditions applies.

7.4.6.4.8 In conditional expressions the symbol'!' is used for logical negation.

7.4.7 SARPs 3.2.6 (AIDC-ASE Protocol Definition)

7.4.7.1 *Protocol Description*

7.4.7.1.1 The protocol description is a specification based on the functional model of the AIDC-ASE.

7.4.7.1.2 The protocol is specified in terms of the actions of the AIDC-ASE in response to the submission of service primitives at its upper service interface, the delivery of service primitives at its lower service interface, and the expiration of internal timers.

7.4.7.1.3 For each AIDC-ASE abstract service primitive or timer event the protocol specifies:

a) the state of the AIDC-ASE in which it is appropriate for the event to occur,

b) the action performed by the AIDC-ASE on the occurrence of the event, and

c) any conditions that need to be tested.

7.4.7.1.4 Through this specification, the protocol enforces the sequencing of AIDC-ASE service primitives and thus also enforces the operational sequencing of AIDC service primitives at the AIDC-AE service boundary.

7.4.7.1.5 Similarly the protocol enforces timing restrictions on the invocation of AIDC-ASE service primitives and thus also enforces the operational timing restrictions that apply to AIDC-AE service primitives invoked by the AIDC-User.

7.4.7.2 *Exception Handling*

7.4.7.2.1 SARPs 3.2.6.2 specifies that in the case of three specific types of exceptional condition the AIDC-ASE should signal AIDC-pvd-abrt.indication with an appropriate value of the AbortReason parameter:

a) when one of the timers listed in SARPs 3.2.6.3 expires;

b) when there is an unspecified irrecoverable error in the AIDC-ASE;

c) when an invalid APDU is received.

7.4.7.2.2 The latter two cases should be considered in the light of SARPs 3.2.6.1.3 which recommends attempting error recovery on the receipt of an unexpected PDU. It may be appropriate to distinguish in an implementation between a delivered AIDC-DATA primitive containing an AIDC Data parameter that can be decoded and has valid contents, but does not contain the correct information type, and one containing an AIDC Data parameter that cannot be decoded validly (the case where an item violates an ASN.1 constraint is discussed below).

7.4.7.2.3 Possible responses to an unexpected PDU include:

a) ignoring the invalid PDU, and thus implicitly relaxing the sequencing rules enforced by the AIDC-ASE, as discussed in 7.4.7.1.4 — this may be especially sensible when the PDU corresponds to a User-confirmation (that is, it is a value of type AIDC-ucf-apdu);

b) issuing a provider abort, in line with SARPs 3.2.6.2.2;

c) if the PDU corresponds to one of the AIDC-ASE primitives that has an Information parameter, responding to the invalid PDU by transmitting a simulated negative User-confirmation (that is, proceeding as if an AIDC-ucf.request primitive had been submitted) with the parameter result having the abstract value 'rejected'.

7.4.7.2.4 However, the last of these options goes outside the SARPs and would only contribute to error recovery if the first option (that is, ignoring the PDU) was selected for unexpected receipt of AIDC-ucf-apdu values by the peer implementation.

7.4.7.2.5 Clause 8.2 of ISO/IEC 8824-3 [18] specifies the default exception handling in case an ASN.1 constraint is violated. According to this specification, this situation should lead to a Provider-abort with diagnostic either 'providererror' or 'undefinederror'; the latter is to be preferred.

7.4.7.3 *Application Timers*

7.4.7.3.1 **Overview**

7.4.7.3.1.1 SARPs Table 3.2.6-1 lists the timers used in the AIDC protocol. It also recommends values for each timer.

7.4.7.3.1.2 In any implementation, all of these timers should be configurable so as to allow for tuning of the timer values in the light of operational experience. It is expected that an agreed set of timer values or ranges will form part of the regional or bilateral agreement between ATSUs when implementing AIDC.

7.4.7.3.1.3 There are likely to be different values for each timer corresponding to two distinct types of operational scenario:

a) a large value for use across oceanic boundaries and continental boundaries of large FIRs with low traffic density;

b) a small value for use across boundaries of small FIRs with high traffic density.

7.4.7.3.1.4 Many of the timers need to take account of round-trip propagation times. It will normally be the case that the communications infrastructure across which the AIDC PDUs are transmitted is under the management of at least two different authorities (that is, the communications providers associated with the two ATSUs, and possibly a transit provider). Since AIDC does not specify an explicit ATSC class (see 7.4.9.2.2), the Internet Communications Service does not have to be configured to meet a specific end-to-end delay.

7.4.7.3.1.5 It can be assumed that transit delay over ground-ground ATN subnetworks and routers is at least as good as ATSC Class H (95% within 50s). In the high density case, ATN communications infrastructure will need to be provided that offers a correspondingly lower transit delay, down to 5 s or less.

7.4.7.3.1.6 The timer values given below are consistent with that used by applications similar to AIDC in use within different ATS regions.

7.4.7.3.1.7 **User Confirmation timer**

7.4.7.3.1.7.1 The User Confirmation timer t_C is used to monitor the receipt of a User-confirmation response, for all AIDC services except Abort services.

7.4.7.3.1.7.2 The value of t_C should allow for

a) generation of the PDU and its transmission to the remote ATSU,

b) reception, decoding and validation of the PDU by the remote ATSU (this may involve retrieving flight data from the flight data processing system in the remote ATSU, if present),

c) generation of the User-confirmation and its transmission back to the originating ATSU, and

d) reception, decoding and validation of the User-confirmation by the originating ATSU.

7.4.7.3.2 A nominal value of t_C suitable for most environments is 2 minutes. A lower value may need to be selected for a high density scenario, down to as little as 12 s.

7.4.7.3.3 **Info-transfer Timers**

7.4.7.3.3.1 The Info-transfer timers t_{1IN} and t_{2IN} monitor the time for transition between an initial invocation of the Info-transfer service primitive in the IDLE state and a subsequent Regime. Thus they control the time for which the underlying association is maintained while the dialogue is idle. (In principle only one of these timers is actually required).

7.4.7.3.3.2 A nominal value for these timers is 10 minutes in the low density/large FIR case and 30 s in the high density/small FIR case.

7.4.7.3.4 **Notifying-Coordinating Timers**

7.4.7.3.4.1 The Notifying-Coordinating timers t_{1NC} and t_{2NC} monitor the time for transition between the Notifying Regime and the Coordinating Regime. Thus they control the time for which the underlying association is maintained while the dialogue is idle. (In principle only one of these timers is actually required).

7.4.7.3.4.2 A nominal value for these timers is 10 minutes in the low density/large FIR case and 30 s in the high density/small FIR case.

7.4.7.3.5 **Response Monitoring Timer**

7.4.7.3.5.1 The Response Monitoring timer t_{1R} monitors the response time for the coordination and transfer primitives following a positive User-confirmation. Thus it detects when an expected response is not received after a positive User-confirmation.

7.4.7.3.5.2 A nominal value for this timer is 10 minutes in the low density/large FIR case and 30 s in the high density/small FIR case.

7.4.7.3.6 **Negative Response Monitoring Timer**

7.4.7.3.6.1 The Negative Response Monitoring timer t_{2R} monitors the response time for the coordination and transfer primitives following a negative User-confirmation. Thus it detects when an expected response is not received after a negative User-confirmation.

7.4.7.3.6.2 A nominal value for this timer is 5 minutes.

7.4.7.3.7 **Transfer Communications Response Monitoring Timer**

7.4.7.3.7.1 The Transfer Communications Response Monitoring timer t_{3R} monitors the response time for the transfer primitives used for transfer of communications following a positive User-confirmation. Thus it detects when an expected response is not received after a positive User-confirmation.

7.4.7.3.7.2 A nominal value for this timer is 5 minutes in the low density/large FIR case and 30 s in the high density/small FIR case.

7.4.7.3.8 **Standby Timer**

7.4.7.3.8.1 The Standby timer t_S extends the time available for a response during coordination negotiation.

7.4.7.3.8.2 A nominal value for this timer is 15 minutes.

7.4.7.3.9 **Coordinating-Transferring Timers**

7.4.7.3.9.1 The Coordinating-Transferring timers t_{1CT} and t_{2CT} monitor the time for transition between the Coordinating Regime and the Transferring Regime. Thus they control the time for which the underlying association is maintained while the dialogue is idle. (In principle only one of these timers is actually required).

7.4.7.3.9.2 A nominal value for these timers is 50 minutes in the low density/large FIR case and 5 minutes in the high density/small FIR case.

7.4.7.3.10 **End Timer**

7.4.7.3.10.1 The End timer t_{TE} monitors the time after completion of the Transferring Regime to the invocation of the AIDC-end service or the commencement of backward coordination. Thus it controls the time for which the underlying association is maintained while the dialogue is idle.

7.4.7.3.10.2 A nominal value for this timer is 100 minutes in the low density/large FIR case and 15 minutes in the high density/small FIR case.

7.4.7.4 ***State Machine Specification***

7.4.7.4.1 As described in 7.4.7.1, the protocol specification is set out in the form of a finite state machine: that is, the protocol machine is modelled as comprising a state which may have one of a finite number of values together with a finite set of events that may occur and a set of actions (including changing the state) that must be performed in response to each event.

7.4.7.4.2 In the AIDC-ASE, the state is structured to make the specification easier to express, by reducing the number of separate states. The state comprises:

a) the main state variable, taking one of the nine values IDLE, NOTIFY, NEGOTIATING, COORDINATED, RE-NEGOTIATING, PRE-TRANSFER, TRANSFERRING, TRANSFERRED, BACKWARD COORDINATING, and

b) a set of auxiliary state variables that define subsets of the main state:

1) vr_1, used to save the type of the last PDU received in an AIDC-DATA.indication;

2) vr_e, used to save the value of the last Result parameter received in an AIDC-crd-end APDU;

3) vs_1, used to save the type of the last PDU sent in an AIDC-DATA.request;

4) vs_e, used to save the value of the last Result parameter sent in an AIDC-crd-end APDU.

7.4.7.4.3 Thus vr_1 and vs_1 can take one of the following values: NULL, notify, coord-start, coord-negot, coord-stndby, coord-end, trns-init, trns-prpsl, trns-comm, trns-assm, trn-start, trns-accept, trns-reject, info-trans, end. Similarly vr_2 and vs_2 can take one of the following values: accepted, rejected.

7.4.7.4.4 For each value of the main state and for each possible event, the SARPs provides a specification of the actions to be performed. Where the action depends on the subsets of the state, different actions are specified conditional on boolean expressions formed from the auxiliary state variables.

7.4.7.4.5 The possible events depend on the current value of the state. When no action for an event is specified in the SARPs for some value of the state, the event is either logically impossible or not allowed by the protocol. In either case, an implementation of the AIDC-ASE is required to treat the occurrence of an event for which no action is specified in the SARPs as an exception according to the procedures in SARPs 3.2.6.2. Exception handling is discussed further in 7.4.7.2.

7.4.7.4.6 The primitives of the AIDC-ASE service are unconfirmed. The state transition following those AIDC-ASE primitives that require a user confirmation (in the form of a response comprising a corresponding AIDC-ucf PDU) occurs after the reception of the user confirmation. Thus, the transition from COORDINATING to COORDINATED takes place when the AIDC-ucf PDU corresponding to the AIDC-crd-end PDU sent by one ATSU is transmitted in the other ATSU and when it is received by the first ATSU.

7.4.7.4.7 In addition the AIDC-ASE state machine uses two auxiliary variables to store information between events. These variables need not be considered as contributing to the state defined above since their values do not affect the type of action to be performed, but merely form input to subsequent actions. These variables are:

a) vr_2, used to save the value of the last Msg Number received in an AIDC-DATA.indication;

b) vs_2, used to save the value of the last Msg Number sent in an AIDC-DATA.request.

7.4.7.4.8 The AIDC-ASE state machine is summarised in the table presented in SARPs 3.2.6.4. The table should be a condensed version of the requirements stated in the preceding text and in case of a conflict the text specification takes precedence.

7.4.7.4.9 In the state table, each cell contains a specification of the actions to be performed for the given event and value of the main state variable. When different actions have to be performed depending on the auxiliary state variables, the cell contents comprises a series of specifications each introduced by a conditional expressions, using the keyword ‘if’, and followed by the actions for that set of conditions, culminating in a default set of actions to be performed if none of the preceding conditions applies.

7.4.7.4.10 In conditional expressions the symbol‘!’ is used for logical negation.

7.4.8 SARPs 3.2.7 (AIDC Formal Definitions)

7.4.8.1 *Abstract Syntax Notation 1*

7.4.8.1.1 The Abstract Syntax Notation 1 (ASN.1) is an International Standard which specifies a notation for abstract syntax definition [17]. Part II.7 gives an overview of the use of ASN.1 to specify ATN protocols.

7.4.8.1.2 ASN.1 is used in the AIDC SARPs to define the AIDC protocol syntax and the information transferred.

7.4.8.1.3 Thus it explicitly specifies the structure of the AIDC application protocol data units.

7.4.8.1.4 In addition, since the information transferred in AIDC APDUs is provided by and delivered through the AIDC-AE and AIDC-ASE abstract services, the ASN.1 definitions implicitly specify the abstract syntax of the parameters of these services, and hence the abstract syntax of the information passed between the AIDC-User and the AIDC-AE.

7.4.8.2 *The AIDC ASN.1*

7.4.8.2.1 Structure

7.4.8.2.1.1 The ASN.1 for the AIDC application is structured as follows:

a) APDU definitions,

b) message definitions,

c) message element definitions, and

d) definitions for error related types.

7.4.8.2.1.2 Where possible, the types reflect the definitions defined for other ATN applications to ensure consistency.

7.4.8.2.1.3 Throughout the ASN.1, extensibility markers (shown as …) have been placed to allow expansion of the AIDC messages and other AIDC types.

7.4.9 SARPs 3.2.8 (Communication Requirements)

7.4.9.1 *Encoding Rules*

7.4.9.1.1 Data items specified using ASN.1 can be encoded systematically using a defined encoding rule for each primitive and constructed data type.

7.4.9.1.2 Several sets of such Encoding Rules have been standardised, of which the most widely used is the Basic Encoding Rules (BER) [20].

7.4.9.1.3 The ATN ULCS SARPs ([2], 4) specify the use of the Packed Encoding Rules (PER) specified in [21]. Part II.7 gives an overview of the use of PER in ATN protocols.

7.4.9.1.4 Encoding using PER results in much smaller encoded PDUs than using BER, but decoding PER-encoded data requires knowledge of the abstract syntax by the decoder, unlike BER.

7.4.9.1.5 PER compilers are available to generate high-level software source code based on the ASN.1 structures defined in 3.2.7 of the SARPs [2].

7.4.9.2 *Quality-of-service Requirements*

7.4.9.2.1 **Overview**

7.4.9.2.1.1 AIDC makes use of the ATN Quality-of-service Parameters Security, Priority, and Residual Error Rate. Guidance on the ATN Quality-of-service Parameters is given in Part II. Specifications for the mapping of application Quality-of-service requirements to the ATN Internet Communications Service (that is, the Transport service) are given in SARPs 4.4.7 [2].

7.4.9.2.2 **Routing Policy**

7.4.9.2.2.1 The Routing Class parameter in the ATN Security Parameter is to be set to the value "ATSC: No Traffic Type Policy Preference" as defined in SARPs 1.3.7 [2]. The requirement for an ATSC routing class follows because AIDC is an ATSC application; there is no requirement for any more detailed traffic type policy because of its ground-ground nature.

7.4.9.2.3 **Priority**

7.4.9.2.3.1 The priority parameter is to be set to the value "normal priority flight safety messages" as defined in SARPs 1.3.8 [2] and specified in SARPs 5.5.2.2. This requirement follows from the specification in Part VI.2.1.4 of [7].

7.4.9.2.4 **Residual Error Rate**

7.4.9.2.4.1 The Residual Error Rate parameter is to be set to the value "low" as defined in SARPs 3.1 [2]. This requirement leads to the use of a checksum at the Transport layer, thereby increasing the integrity of AIDC exchanges; integrity is important because the use of PER means that a small transmission error in one PDU can cause the application to decode invalid data from the rest of the PDU and such an error will only accidentally be detectable (for instance, if the decoded invalid data exceeds constraints specified in the syntax).

7.4.10 SARPs 3.2.9 (AIDC-User Requirements)

7.4.10.1 *Overview*

7.4.10.1.1 This section of the SARPs is necessary to provide a complete the set of requirements ensuring that implementations of AIDC are both interoperable and operationally acceptable.

7.4.10.2 *Interoperability*

7.4.10.2.1 The preceding SARPs sections specify the AIDC service and protocol. In the course of this specification, several options are identified that need to be mutually decided as part of a regional or bilateral agreement. These decisions will be based on the operational requirements that apply in each case: for instance, different timer values will be chosen for an oceanic FIR boundary compared with a boundary in continental airspace such as Central Europe or South-East Asia. SARPs 3.2.9.1 lists the aspects that must be covered by such an agreement. These aspects have been noted in more detail throughout the description of the SARPs in this chapter.

7.4.10.2.2 To assist in establishing such an agreement as part of a regional implementation, States may consider developing a pro forma, listing the required choices and acceptable ranges of values, that may be annexed to the agreement.

7.4.10.3 *Message Handling*

7.4.10.3.1 The requirements in SARPs 3.2.9.2.1 [2] are necessary to ensure that the operational needs of the AIDC messages that can be sent using the Info-transfer service primitive are not compromised by processing in the AIDC-User.

7.4.10.4 *Operational Timers*

7.4.10.4.1 The AIDC-User is required to maintain timers in order to ensure that information required through the human computer interface is provided before the expiration of protocol timers, or for other operational reasons.

7.4.10.4.2 The Standby Timer will normally be set to a value less than that of the technical timer t_S.

7.4.10.5 *AIDC-AE Specific Requirements*

7.4.10.5.1 **Management of AIDC-AE Instantiations**

7.4.10.5.1.1 For each AIDC dialogue, there will be a thread of control in the AIDC-User software and a corresponding instantiation of the AIDC-AE. How AIDC-AE instantiations are distinguished is an implementation issue, but it may be sensible to consider using operating system facilities for processes or threads (that is, lightweight processes within a single address space) to support the multiple instantiations of the AIDC-AE.

7.4.11 SARPs 3.2.10 (Sequence Diagrams)

7.4.11.1 Strictly the sequence diagrams provided in SARPs 3.2.10 do not add anything to the preceding specifications. However, they are included in the SARPs because they have proven to add significantly to the comprehensibility of the specification. Implementors are recommended to study these diagrams closely to ensure that the cases illustrated are correctly handled. In addition, the sequence diagrams may be useful as input to the definition of a test specification.

7.5 Dimensions

7.5.1 PDU Size

7.5.1.1 There is nothing in the AIDC SARPs that restricts the contents of the AIDC messages which can be generated, except those data elements which are optional. In this context optional means that the generating system does not need to provide default value(s) when it has no knowledge of the information.

7.5.1.2 Typical sizes of some example AIDC PDUs are indicated in Table 7.5-1. The examples have been chosen to be representative of the different types used in AIDC.

7.5.1.3 The aidc-ucf-apdu is shown since it is frequently used.

7.5.1.4 The aidc-nfy-apdu is typical of the APDUs that represent Coordination messages (aidc-crd-start-apdu, aidc-crd-end-apdu, aidc-crd-ngtt-apdu, aidc-crd-stndby-apdu). The dominating component in the maximum size is the Route which comprises a SEQUENCE (0..128) of RouteInformation items, the encoding for each of which may be some 60 octets in length. It is also representative of the aidc-inf-tfr-apdu for the case of GeneralPoint.

7.5.1.5 The aidc-tfr-cntrl-req-apdu is typical of the APDUs that represent Transfer messages (aidc-tfr-init-apdu, aidc-tfr-rqst-apdu, aidc-tfr-prpsl-apdu, aidc-tfr-accept-apdu, aidc-tfr-cntrl-rsp-apdu, aidc-tfr-comm-apdu, aidc-tfr-comm-assm-apdu). It is also representative of aidc-inf-tfr-apdu for the case of GeneralExecutiveData and SurveillanceGeneral.

7.5.1.6 For the aidc-inf-tfr-apdu in the case of GeneralFreeText and EmergencyFreeText, the maximum size is some 290 octets, depending on the length of the text to be transferred.

Table 7.5-1. AIDC PDU Sizes

AIDC PDU	Typical Size in octets	Maximum Size in octets
AIDC Notify message (aidc-nfy-apdu)	280	8000
AIDC User Confirmation message (aidc-ucf-apdu)	3	37
AIDC TransferControl message (aidc-tfr-cntrl-req-apdu)	30	45

7.5.2 Rate of AIDC Message Transmission

7.5.2.1 For a pair of ATSUs, the rate at which AIDC messages are transmitted is closely related to the number of aircraft approaching the common FIR boundary within a certain distance and/or time limit or crossing this boundary.

7.5.2.2 This number of aircraft highly depends on the prevailing operational environment on both sides of the FIR boundary and, to a certain extent, on the degree of automation of both ATSUs.

7.5.2.3 As noted in 7.15.6, the set of AIDC messages, and consequently their number, to be exchanged on a flight by flight basis is subject to bilateral conventions which also depend on the operational environment.

7.5.2.4 Assuming that the minimum subset of AIDC messages defined in 7.1.5.6 is used, the total number of messages exchanged on a flight per flight basis is typically four without taking into account the user confirmation APDUs.

7.5.3 Dialogue and connection requirements

7.5.3.1 As mentioned in 7.2.7.3, in version 1 of the AIDC SARPs [2], the maximum number of concurrent transport connections needs to be considered when sizing an AIDC implementation; this should be at least the maximum number of flights for which the ATSU may need to maintain dialogues with all other ATSUs.

7.6 Indexes and Tables

7.6.1 AIDC Data Dictionary

7.6.1.1 Table 7.6-1 lists the data items defined in Part VI.4 of [7] for AIDC together with their definitions and the identifiers of the corresponding ASN.1 types in the ASN.1 module AIDCMessageSetVersion1, specified in SARPs 3.2.7.

Table 7.6-1. AIDC Data Dictionary

Item	Definition	ASN.1 type
Aircraft address	A unique combination of 24 bits available for assignment to an aircraft for the purpose of air-ground communications, navigation and surveillance.	AircraftAddress ::= BIT STRING (SIZE(24))
Aircraft Identification	A group of letters, figures or a combination thereof which is identical to or the code equivalent of the aircraft callsign. It is used in Field 7 of the ICAO Model flight plan.	AircraftIdentification ::= IA5String (SIZE(2..7))
Aircraft Type	An IA5 string consisting of 2-5 characters used to identify the aircraft type as defined in *Aircraft Type Designators* (ICAO Doc 8643). The aircraft type can be optionally preceded by the number of aircraft if more than one.	AircraftType ::= IA5String (SIZE(2..4))
ATS Route Identifier	Specifies the particular airway or standard instrument departure (SID) to be used within the route of the aircraft.	ATSRouteDesigtnator::= IA5String (SIZE(2..7))

ATW Level	Contains *ATW Level Tolerance* and *Level*.	ATWLevel ::= SEQUENCE
		{
		atw [0] ATWLevelTolerance,
		level [1] Level
		}

(_)ATW Level Tolerance	Indicates the vertical tolerance factor for level clearances. Used in level clearances to indicate the acceptable vertical clearance of an aircraft relative to a particular level. Indicates:
	a) at,
	b) at or above, or
	c) at or below.
		ATWLevelTolerance ::= ENUMERATED
		{
		at (0),
		atorabove (1),
		atorbelow (2)
		}

(_)Boundary Estimate Data	Specifies information related to the boundary crossing. Boundary fix consists of the following sequence of information:
	a) Boundary fix,
	b) Crossing time, and
	c) Crossing level.
		BoundaryEstimate ::= SEQUENCE
		{
		boundaryFix [0] Position,
		crossingTime [1] Time,
		crossingLevel [2] Level,
		atwLevel [3] ATWLevel OPTIONAL
		}

Item	**Definition**	**ASN.1 type**
Boundary fix	Specifies the fix at the boundary between the two ATSUs as a Position.	See Position
Code (SSR)	Specifies the Mode A value for the aircraft.	BeaconCode ::= SEQUENCE SIZE (4) OF BeaconCodeOctalDigit BeaconCodeOctalDigit ::= INTEGER (0..7)

(_)CNS Equipment	Component variable used to indicate the type of communication, navigation and approach aid equipment on an aircraft as defined in the *Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services* (PANS-RAC, ICAO Doc 4444). Aircraft equipment code consists of the following sequence of information:
	a) an indication of whether or not *COM NAV Approach Equipment* is available,
	b) optionally a sequence containing 1-24 *COM NAV Equipment Status*, and
	c) an indication of what *SSR Equipment* is available.
		CNSEquipment ::= SEQUENCE
		{
		SEQUENCE SIZE (0..24) OF comNavEquipmentStatus [0] ComNavEquipmentStatus OPTIONAL,
		ssrEquipmentAvailable [1] SSREquipmentAvailable,
		adsAvailable [2] BOOLEAN,
		acasAvailable [3] BOOLEAN,
		dataLink [4] SEQUENCE SIZE (0..4) OF DataLink
		}

COM NAV Approach Equipment Available	Indicates the presence or absence of communication, navigation and approach aid equipment.	Indicated by comNavEquipmentStatus

COM NAV Equipment Status	Indicates which communication, navigation and approach aid equipment is available for use as defined in the *Procedures for Air Navigation Services - Rules of the Air and Air Traffic Services* (PANS-RAC, ICAO Doc 4444).	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),
		...
		}

Item	Definition	ASN.1 type
Crossing Level	Specifies the Boundary Crossing Level in Level.	See Level
Crossing Time	Specifies the Boundary Crossing Time in Time.	See Time
Degree Minutes	Provides minutes of a latitude or longitude degree.	DegreeMinutes ::= INTEGER (0..5999)
Degree Seconds	Provides seconds of a latitude or longitude degree.	DegreeSeconds ::= INTEGER (0..59)

Departure Aerodrome	Flight plan departure aerodrome.	DepartureAirportTime ::= SEQUENCE
		{
		airport [0] Airport,
		time [1] Time OPTIONAL
		}

Departure Time	Specifies the time of Departure in Time	See Departure Aerodrome
Destination Aerodrome	Flight plan destination aerodrome.	DestinationAirport ::= Airport

Direct Routing	A sequence of Next Way Point and Following Way Point.	DirectRouting ::= SEQUENCE
		{
		fix2 [0] Position OPTIONAL,
		fix1 [1] Position
		}

Distance	Provides the distance in SI or non-SI units.	Distance ::= CHOICE
		{
		distanceNM [0] DistanceNM,
		distancekm [1] Distancekm
		}

Error Code	Specifies the type of error found in a received message.	**ErrorCode** ::= ENUMERATED { invalidNumberOfAircraft (0), invalidAircraftType (1), invalidWakeTurbulenceCategory (2), invalidBeaconCodeOctalDigit (3), invalidFixName (4), invalidNavaid (5), invalidAirport (6), invalidLatitude (7), invalidLongitude (8), invalidTime (9), invalidLevelFeet (10), invalidLevelMeters (11), invalidLevelFlightLevel (12), invalidLevelFlightLevelMetric (13), invalidATWLevelTolerance (14), invalidComNavEquipmentStatus (15), invalidSSREquipmentAvailable (16), invalidDataLink (17), invalidSpeedGround (18), invalidSpeedGroundMetric (19), invalidSpeedMach (20), invalidSpeedIndicated (21), invalidSpeedIndicatedMetric (22),

Item	Definition	ASN.1 type
		invalidSpeedTrue (23), invalidSpeedTrueMetric (24), invalidVerticalDirection (25), invalidVerticalRateEnglish (26), invalidVerticalRateMetric (27), invalidAircraftIdentification (28), invalidSelcal (29), invalidRegistration (30), invalidAirframeID (31), invalidFlightRule (32), invalidFlightType (33), invalidFrequencyHF (34), invalidFrequencyVHFChannel (35), invalidFrequencyUHF (36), invalidFrequencySatChannel (37), invalidFunctionalAddress (38), invalidReleaseIndicator (39), invalidDistanceKm (40), invalidDistanceNM (41), invalidATSRouteDesignator (42), invalidTrackName (43), invalidmsgnumber (250), invalidreferenceid (251), invalidcallingICAOFacilityDesignation (252), invalidcalledICAOFacilityDesignation (253), invalidtimestamp (254), unknown (255), ... }
Error Data	Specifies the erroneous data found in a received message	ErrorData ::= BIT STRING (SIZE(1..256))

(_)Executive Data	Specifies the intended clearance to be passed to the next controller that will be in charge of the aircraft. Consists of one or more of the following:
	a) *Speed*,
	b) *Level*,
	c) *Heading*,
	d) *Vertical rate*,
	e) *Direct routing*, or
	f) *Current cleared level*.
		ExecutiveData ::= SEQUENCE
		{
		speed [0] Speed OPTIONAL,
		level [1] Level OPTIONAL,
		heading [2] DegreesMagnetic OPTIONAL,
		vertRate [3] VerticalChange OPTIONAL,
		directRouting [4] DirectRouting OPTIONAL
		}

Fix Name	Specifies the ICAO identifier for a given fix.	FixName ::= IA5String (SIZE(1..5))
Flight Level	As defined in PANS/RAC (Doc 4444).	See Level

Item	Definition	ASN.1 type

(_)Flight Rules	Specifies the flight rule. Can be one of the following:
	a) *IFR*,
	b) *VFR*,
	c) *IFR first* or
	d) *VFR first.*
		FlightRule ::= ENUMERATED
		{
		ifr (0),
		vfr (1),
		ifrfirst (2),
		vfrfirst (3)
		}

Following Way Point	Indicates the way point after the next way point as a Position	See Position
Free Text	Used to convey unstructured information.	FreeText ::= IA5String (SIZE(1..256))

(_)Frequency	Specifies the frequency and an indicator of the RF spectrum used for the given frequency. The types of frequency that can be provided include:
	a) *Frequency HF*,
	b) *Frequency VHF Channel*,
	c) *Frequency UHF*, or
	d) *Frequency Sat Channel*.
		Frequency ::= CHOICE
		{
		frequencyHF [0] FrequencyHF,
		frequencyVHFChannel [1] FrequencyVHFChannel,
		frequencyUHF [2] FrequencyUHF,
		frequencySatChannel [3] FrequencySatChannel
		}

FrequencyHF		FrequencyHF ::= INTEGER (2850..28000) -- unit = kilohertz, Range (2850..28000), resolution = 1
FrequencyVHFChannel		FrequencyVHFChannel ::= INTEGER (23600..27398) -- unit = VHF Channel, range (118.000..136.990), resolution = 0.005
FrequencyUHF		FrequencyUHF ::= INTEGER (9000..15999) -- unit = megahertz, Range (225.000..399.975), resolution = 0.025
Frequency Sat Channel	Frequency Sat Channel corresponds to a 12-digit telephone number	FrequencySatChannel ::= NumericString (SIZE(12))
Functional Address	Specifies an identifier used to direct the message to a locally defined position used instead of associating the message with a flight.	FunctionalAddress ::= IA5String (SIZE (1..18))

Heading	Provides aircraft heading in degrees	Degrees ::= CHOICE
		{
		degreesMagnetic [0] DegreesMagnetic,
		degreesTrue [1] DegreesTrue
		}
		
		DegreesMagnetic ::= INTEGER (1..360)
		-- unit = degree, Range (1..360), resolution = 1
		DegreeMinutes ::= INTEGER (0..5999)
		-- unit = Minute, Range (0..59.99), resolution = 0.01

Hours	Specifies the hour in 24-hour notation.	TimeHours ::= INTEGER (0..23)

Item	Definition	ASN.1 type

Latitude	Provides latitude as *Latitude Degrees*, *Degree Minutes* (optional), *Degree Seconds* (optional) and the *Direction*.	Latitude ::= SEQUENCE
		{
		latitudeDegrees [0] LatitudeDegrees,
		latitudeMinutes [1] DegreeMinutes OPTIONAL,
		latitudeSeconds [2] DegreeSeconds OPTIONAL,
		latitudeDirection [3] LatitudeDirection
		}
		
		LatitudeDirection ::= ENUMERATED
		{
		north (0),
		south (1)
		}

Latitude Degrees	Degrees of latitude.	LatitudeDegrees ::= INTEGER (0..90000)

Latitude Longitude	Sequence of *Latitude* and *Longitude*.	LatitudeLongitude ::= SEQUENCE
		{
		latitude [0] Latitude,
		longitude [1] Longitude
		}

Level	Specifies the level in one of the following ways:
	a) *Level* in metre or feet,
	b) *Flight Level* in metre or feet.
		Level ::= CHOICE
		{
		levelFeet [0] LevelFeet,
		levelMetre [1] LevelMetre,
		levelFlightLevel [2] LevelFlightLevel,
		levelFlightLevelMetric [3] LevelFlightLevelMetric
		}
		
		LevelFeet ::= INTEGER (-60..7000)
		-- unit = Feet, Range (-600..70000), resolution = 10
		
		LevelMetre ::= INTEGER (-30..25000)
		-- unit = metre, Range (-30..25000), resolution = 1
		
		LevelFlightLevel ::= INTEGER (30..700)
		-- unit = Level (100 feet), Range (30..700), resolution = 1
		
		LevelFlightLevelMetric ::= INTEGER (100..2500)
		-- unit = Level (10 metre), Range (100..2500), resolution = 1

Level Current	Specifies the current aircraft level.	See Level

Longitude	Provides longitude as *Longitude Degrees*, *Degree Minutes (optional), Degree Seconds (optional)* and *Direction*.	Longitude ::=SEQUENCE
		{
		longitudeDegrees [0] LongitudeDegrees,
		longitudeMinutes [1] DegreeMinutes OPTIONAL,
		longitudeSeconds [2] DegreeSeconds OPTIONAL,
		longitudeDirection [3] LongitudeDirection
		}
		LongitudeDirection ::= ENUMERATED
		{
		east (0),
		west (1)
		}

Longitude Degrees	Degrees of longitude.	LongitudeDegrees ::= INTEGER (0..180000)

Item	**Definition**	**ASN.1 type**
Next Way Point	Specifies the next way point as a Position	See Position
Number of aircraft	Number of aircraft concerned by the message (as in the ICAO flight plan)	NumberOfAircraft ::= INTEGER (1..2)
Other Information	Used to provide additional information in Free Text.	OtherInformation ::= FreeText

Place Bearing	Sequence of *Fix Name, Latitude Longitude* (optional), and *Degrees*.	PlaceBearing ::= SEQUENCE
		{
		fixName [0] FixName,
		latitudeLongitude [1] LatitudeLongitude OPTIONAL,
		degrees [2] Degrees
		}

Place Bearing Distance	Used to indicate a location based on the degrees and distance from a known point. Provided using *Place Bearing* and *Distance* data.	PlaceBearingDistance ::= SEQUENCE
		{
		placeBearing [0] PlaceBearing,
		distance [1] Distance
		}

Place Bearing Place Bearing	Used to define a point as the intersection formed by two bearings from two known points. Provided as two *Place Bearing*.	PlaceBearingPlaceBearing ::= SEQUENCE SIZE (2) OF PlaceBearing

(_)Position	Information used to specify a location. Position can be specified as:
	a) *Fix Name*,
	b) *Navaid*,
	c) *Aerodrome*,
	d) *Latitude Longitude*, or
	e) *Place Bearing Distance*.
		Position ::= CHOICE
		{
		fixName [0] FixName,
		navaid [1] Navaid,
		airport [2] Airport,
		latitudeLongitude [3] LatitudeLongitude,
		placeBearingDistance [4] PlaceBearingDistance
		}

Published Identifier	Used to provide the location of the specified fix. Provided using *Fix Name* and *Latitude Longitude*.	PublishedIdentifier ::= SEQUENCE
		{
		fixName [0] FixName,
		latitudeLongitude [1] LatitudeLongitude
		}

Registration	Provides the aircraft registration	Registration ::= IA5String (SIZE(7))

ReleaseIndicator	Indicates the degree of freedom allowed to the R-ATSU for reclearing a transferred flight while it remains in the area of common interest.	ReleaseIndicator ::= ENUMERATED
		{
		climb (0),
		descent (1),
		turns (2),
		allActions (3)
		}

Item	**Definition**	**ASN.1 type**

Route	Specifies route of flight using *Route Information*	Route ::= SEQUENCE
		{
		SEQUENCE SIZE (1..128) OF
		RouteInformation,
		position [0] Position,
		time [1] Time,
		level [2] Level,
		speedGround [3] SpeedGround,
		trueTrackAngle [4] TrueTrackAngle
		}

(_)Route Information	Indicate the method used to define the aircraft route of flight. The actual aircraft route of flight will probably consist of multiple *Route Information* sequences as follows:
	a) *Published Identifier* (optional)
	b) *Latitude Longitude* (optional),
	c) *Place Bearing Place Bearing* (optional),
	d) *Place Bearing Distance* (optional),
	e) *ATS Route Identifier* (optional), and
	f) *Track Detail* (optional).
		RouteInformation ::= CHOICE
		{
		publishedIdentifier [0] PublishedIdentifier,
		latitudeLongitude [1] LatitudeLongitude,
		placeBearingPlaceBearing [2] PlaceBearingPlaceBearing,
		placeBearingDistance [3] PlaceBearingDistance,
		aTSRouteDesignatorr [4] ATSRouteDesignator,
		trackDetail [5] TrackDetail
		}

Selcal	Provides the Selcal of the aircraft	Selcal ::= IA5String (SIZE(4))

(_)Speed	Provides the aircraft speed as one of the following:
	a) *Speed Indicated*,
	b) *Speed True*,
	c) *Speed Ground*, or
	d) *Speed Mach*.
		Speed ::= CHOICE
		{
		speedGround [0] SpeedGround,
		speedGroundMetric [1] SpeedGroundMetric,
		speedMach [2] SpeedMach,
		speedIndicated [3] SpeedIndicated,
		speedIndicatedMetric [4] SpeedIndicatedMetric,
		speedTrue [5] SpeedTrue,
		speedTrueMetric [6] SpeedTrueMetric
		}

Speed Ground	Ground speed expressed in either English or metric units.	SpeedGround ::= INTEGER (-50..2000) -- unit = Knots, Range (-50..2000), resolution = 1 SpeedGroundMetric ::= INTEGER (-100..4000) -- unit = kilometre/hour, Range (-100..4000), resolution = 1
Speed Indicated	Indicated aircraft speed expressed in either English or metric units.	SpeedIndicated ::= INTEGER (0..400) -- unit = Knots, Range (0..400), resolution = 1 SpeedIndicatedMetric ::= INTEGER (0..800) -- unit = kilometre/hour, Range (0..800), resolution = 1
Speed Mach	Aircraft speed specified as a Mach value.	SpeedMach ::= INTEGER (500..4000) -- unit = Mach, Range (0.5..4.0), resolution = 0.001

Item	Definition	ASN.1 type
Speed True	Aircraft true speed expressed in either English or metric units.	SpeedTrue ::= INTEGER (0..2000) -- unit = Knots, Range (0..2000), resolution = 1 SpeedTrueMetric ::= INTEGER (0..4000) -- unit = kilometre/hour, Range (0..4000), resolution = 1

SSR Equipment Available	Indicates which surveillance equipment is available for use as defined in the *Procedures for Air Navigation Services — Rules of the Air and Air Traffic Services* (PANS-RAC, ICAO Doc 4444).	SSREquipmentAvailable ::= ENUMERATED
		{
		nnil (0),
		atransponderModeA (1),
		ctransponderModeAandC (2),
		xatransponderModeS (3),
		ptransponderModeSPA (4),
		itransponderModeSID (5),
		satransponderModeSPAID (6),
		...
		}
		-- PA: Pressure Level; ID: Aircraft Identification

Time	Sequence of *Hours* and *Time Minutes*.	Time ::= SEQUENCE
		{
		hours [0] TimeHours,
		minutes [1] TimeMinutes
		}

Time Minutes	Specifies time in minutes of an hour.	TimeMinutes ::= INTEGER (0..59)

(_)Track Data	Specifies the current position of the aircraft. Contains the following:
	a) *Position,*
	b) *Time,*
	c) *Level (optional),*
	d) *Speed Ground (optional), or*
	e) *True Track Angle (optional).*
		TrackData ::= SEQUENCE
		{
		position [0] Position,
		time [1] Time,
		level [2] Level,
		speedGround [3] SpeedGround,
		trueTrackAngle [4] TrueTrackAngle
		}

Track Detail	Associates a sequence of fixes with a particular track name. Specified using *Track Name* and *Latitude Longitude.*	TrackDetail ::= SEQUENCE
		{
		trackName [0] TrackName,
		latitudeLongitude [1] LatitudeLongitude
		}

Track Name	Specifies the name of an identified group of points which make up a section of a route.	TrackName ::= IA5String (SIZE(1..6))
True Track Angle	Specifies true track angle to the next way-point using degrees.	TrueTrackAngle ::= Degrees

Item	Definition	ASN.1 type

(_)Type of flight	Specifies the type of flight. Can be one of the following:
	a) *Scheduled Air Transport*,
	b) *Non Scheduled Air Transport*,
	c) *General Aviation*,
	d) *Military*, and
	e) *Other flights*.
		FlightType ::= ENUMERATED
		{
		scheduledAirTransport (0),
		nonScheduledAirTransport (1),
		generalAviation (2),
		military (3),
		otherFlights (4)
		}

Vertical Rate	Rate of climb/descent (climb positive, descent negative)	VerticalRate ::= CHOICE
		{
		verticalRateEnglish [0] VerticalRateEnglish,
		verticalRateMetric [1] VerticalRateMetric
		}

Wake Turbulence category	1 character used to specify the wake turbulence category.	WakeTurbulenceCategory ::= ENUMERATED
		{
		high (0),
		medium (1),
		low (2)
		}

7.6.2 ASN.1 Index

7.6.2.1 Table 7.6-2 lists the ASN.1 identifiers used in the ASN.1 module AIDCMessageSetVersion1 specified in SARPs 3.2.7, showing the constructed types to which they belong, if any.

Table 7.6-2. ASN.1 Identifiers in Module AIDCMessageSetVersion1

Identifier	In type	Constructor	Tag/Size	Of type
_Route_1	Route	SEQUENCE	SIZE (1..128)	SEQUENCE OF RouteInformation
acasAvailable	CNSEquipment	SEQUENCE	[3]	BOOLEAN
accepted	Result	ENUMERATED	0	
adsAvailable	CNSEquipment	SEQUENCE	[2]	BOOLEAN
aidc-crd-end-apdu	AIDC-APDU	CHOICE	[3]	AIDC-crd-end-apdu
aidc-crd-end-apdu	MessageType	ENUMERATED	-3	
aidc-crd-ngtt-apdu	AIDC-APDU	CHOICE	[4]	AIDC-crd-ngtt-apdu
aidc-crd-ngtt-apdu	MessageType	ENUMERATED	-4	
aidc-crd-start-apdu	AIDC-APDU	CHOICE	[2]	AIDC-crd-start-apdu
aidc-crd-start-apdu	MessageType	ENUMERATED	-2	
aidc-crd-stndby-apdu	AIDC-APDU	CHOICE	[5]	AIDC-crd-stndby-apdu
aidc-crd-stndby-apdu	MessageType	ENUMERATED	-5	
aidc-end-apdu	AIDC-APDU	CHOICE	[15]	AIDC-end-apdu
aidc-end-apdu	MessageType	ENUMERATED	-15	
aidc-inf-tfr-apdu	AIDC-APDU	CHOICE	[14]	AIDC-inf-tfr-apdu
aidc-inf-tfr-apdu	MessageType	ENUMERATED	-14	
aidc-nfy-apdu	AIDC-APDU	CHOICE	[1]	AIDC-nfy-apdu
aidc-nfy-apdu	MessageType	ENUMERATED	-1	

Identifier	In type	Constructor	Tag/Size	Of type
aidc-tfr-accept-apdu	AIDC-APDU	CHOICE	[9]	AIDC-tfr-accept-apdu
aidc-tfr-accept-apdu	MessageType	ENUMERATED	-9	
aidc-tfr-cntrl-req-apdu	AIDC-APDU	CHOICE	[10]	AIDC-tfr-cntrl-Req-apdu
aidc-tfr-cntrl-req-apdu	MessageType	ENUMERATED	-10	
aidc-tfr-cntrl-rsp-apdu	AIDC-APDU	CHOICE	[11]	AIDC-tfr-cntrl-Rsp-apdu
aidc-tfr-cntrl-rsp-apdu	MessageType	ENUMERATED	-11	
aidc-tfr-comm-apdu	AIDC-APDU	CHOICE	[12]	AIDC-tfr-comm-apdu
aidc-tfr-comm-apdu	MessageType	ENUMERATED	-12	
aidc-tfr-comm-assm-apdu	AIDC-APDU	CHOICE	[13]	AIDC-tfr-comm-assm-apdu
aidc-tfr-comm-assm-apdu	MessageType	ENUMERATED	-13	
aidc-tfr-init-apdu	AIDC-APDU	CHOICE	[6]	AIDC-tfr-init-apdu
aidc-tfr-init-apdu	MessageType	ENUMERATED	-6	
aidc-tfr-prpsl-apdu	AIDC-APDU	CHOICE	[8]	AIDC-tfr-prpsl-apdu
aidc-tfr-prpsl-apdu	MessageType	ENUMERATED	-8	
aidc-tfr-rqst-apdu	AIDC-APDU	CHOICE	[7]	AIDC-tfr-rqst-apdu
aidc-tfr-rqst-apdu	MessageType	ENUMERATED	-7	
aidc-ucf-apdu	AIDC-APDU	CHOICE	[0]	AIDC-ucf-apdu
aidc-ucf-apdu	MessageType	ENUMERATED	0	
AircraftAddress	AircraftAddress	none	(SIZE(24))	BIT STRING
AircraftIdentification	AircraftIdentification	none	(SIZE(2..7))	IA5String
aircraftIdentification	FlightID	SEQUENCE	[0]	AircraftIdentification
aircraftNumberType	CoordinateInitial	SEQUENCE	[5]	AircraftNumberType
aircraftNumberType	GeneralPoint	SEQUENCE	[6]	AircraftNumberType
aircraftNumberType	Notify	SEQUENCE	[5]	AircraftNumberType
aircraftType	AircraftNumberType	SEQUENCE	[1]	AircraftType
AircraftType	AircraftType	none	(SIZE(2..4))	IA5String
airframeID	FlightID	SEQUENCE	[3] OPTIONAL	AircraftAddress
Airport	Airport	none	(SIZE(4))	IA5String
airport	DepartureAirportTime	SEQUENCE	[0]	Airport
airport	Position	CHOICE	[2]	Airport
allActions	ReleaseIndicator	ENUMERATED	-3	
aloranA	ComNavEquipmentStatus	ENUMERATED	0	
at	ATWLevelTolerance	ENUMERATED	0	
atorabove	ATWLevelTolerance	ENUMERATED	-1	
atorbelow	ATWLevelTolerance	ENUMERATED	-2	
atransponderModeA	SSREquipmentAvailable	ENUMERATED	-1	
ATSRouteDesignator	ATSRouteDesignator	none	(SIZE(2..7))	IA5String
aTSRouteDesignator	RouteInformation	CHOICE	[4]	ATSRouteDesignator
atw	ATWLevel	SEQUENCE	[0]	ATWLevelTolerance
atwLevel	BoundaryEstimate	SEQUENCE	[3] OPTIONAL	ATWLevel
BeaconCode		SEQUENCE OF	SIZE (4)	BeaconCodeOctalDigit
beaconCode	CoordinateInitial	SEQUENCE	[4] OPTIONAL	BeaconCode
beaconCode	CoordinateUpdate	SEQUENCE	[3] OPTIONAL	BeaconCode
beaconCode	GeneralPoint	SEQUENCE	[5] OPTIONAL	BeaconCode
beaconCode	Notify	SEQUENCE	[4] OPTIONAL	BeaconCode
BeaconCodeOctalDigit	BeaconCodeOctalDigit	none	(0..7)	INTEGER
boundaryEstimate	Cancel	SEQUENCE	[3] OPTIONAL	BoundaryEstimate

Identifier	In type	Constructor	Tag/Size	Of type
boundaryEstimate	CoordinateInitial	SEQUENCE	[7]	BoundaryEstimate
boundaryEstimate	CoordinateNegotiate	SEQUENCE	[3]	BoundaryEstimate
boundaryEstimate	CoordinateUpdate	SEQUENCE	[4]	BoundaryEstimate
boundaryEstimate	GeneralPoint	SEQUENCE	[8] OPTIONAL	BoundaryEstimate
boundaryEstimate	Notify	SEQUENCE	[7]	BoundaryEstimate
boundaryFix	BoundaryEstimate	SEQUENCE	[0]	Position
calledICAOFacilityDesignation	AIDC-crd-start-apdu	SEQUENCE	[0]	ICAOFacilityDesignation
calledICAOFacilityDesignation	AIDC-nfy-apdu	SEQUENCE	[0]	ICAOFacilityDesignation
callingICAOFacilityDesignation	AIDC-crd-start-apdu	SEQUENCE	[1] OPTIONAL	ICAOFacilityDesignation
callingICAOFacilityDesignation	AIDC-nfy-apdu	SEQUENCE	[1] OPTIONAL	ICAOFacilityDesignation
cancel	AIDC-end-apdu	SEQUENCE	[0]	Cancel OPTIONAL
climb	ReleaseIndicator	ENUMERATED	0	
cloranC	ComNavEquipmentStatus	ENUMERATED	-1	
cnsEquipment	CoordinateInitial	SEQUENCE	[6] OPTIONAL	CNSEquipment
cnsEquipment	GeneralPoint	SEQUENCE	[7]	CNSEquipment
cnsEquipment	Notify	SEQUENCE	[6] OPTIONAL	CNSEquipment
comNavEquipmentStatus	CNSEquipment	SEQUENCE	[0]	SEQUENCE SIZE (0..24) OF ComNavEquipmentStatus
componentType	ApplicationErrorData	SEQUENCE	[1]	ComponentType
coordinateaccept	Enddata	CHOICE	[0]	CoordinateAccept
coordinateinitial	Startdata	CHOICE	[0]	CoordinateInitial
coordinatenegotiate	AIDC-crd-ngtt-apdu	SEQUENCE	[0]	CoordinateNegotiate
coordinatereject	Enddata	CHOICE	[1]	CoordinateReject
coordinatestandby	AIDC-crd-stndby-apdu	SEQUENCE	[0]	CoordinateStandby
coordinateupdate	Startdata	CHOICE	[1]	CoordinateUpdate
crossingLevel	BoundaryEstimate	SEQUENCE	[2]	Level
crossingTime	BoundaryEstimate	SEQUENCE	[1]	Time
ctAircraftNumberType	ComponentType	ENUMERATED	-2	
ctBeaconCode	ComponentType	ENUMERATED	-3	
ctBoundaryEstimate	ComponentType	ENUMERATED	-4	
ctCNSEquipment	ComponentType	ENUMERATED	-5	
ctDepartureAirportTime	ComponentType	ENUMERATED	-6	
ctDestinationAirport	ComponentType	ENUMERATED	-7	
ctExecutiveData	ComponentType	ENUMERATED	-8	
ctFlightID	ComponentType	ENUMERATED	-9	
ctFlightRuleFlightType	ComponentType	ENUMERATED	-10	
ctFreeText	ComponentType	ENUMERATED	-11	
ctFrequency	ComponentType	ENUMERATED	-12	
ctFunctionalAddress	ComponentType	ENUMERATED	-13	
ctNotApplicable	ComponentType	ENUMERATED	-1	
ctransponderModeAandC	SSREquipmentAvailable	ENUMERATED	-2	
ctReleaseIndicator	ComponentType	ENUMERATED	-14	
ctRoute	ComponentType	ENUMERATED	-15	
ctTrackData	ComponentType	ENUMERATED	-16	
ctUnknown	ComponentType	ENUMERATED	0	
ctUnrecognised	ComponentType	ENUMERATED	-255	
dataLink	CNSEquipment	SEQUENCE	[4]	SEQUENCE SIZE (0..4) OF DataLink

Identifier	In type	Constructor	Tag/Size	Of type
date	YMDHMS	SEQUENCE	[0]	Date
day	Date	SEQUENCE	[2]	Day
Day	Day	none	(1..31)	INTEGER
ddme	ComNavEquipmentStatus	ENUMERATED	-2	
DegreeMinutes	DegreeMinutes	none	(0..5999)	INTEGER
degrees	PlaceBearing	SEQUENCE	[2]	Degrees
DegreeSeconds	DegreeSeconds	none	(0..59)	INTEGER
degreesMagnetic	Degrees	CHOICE	[0]	DegreesMagnetic
DegreesMagnetic	DegreesMagnetic	none	(1..360)	INTEGER
degreesTrue	Degrees	CHOICE	[1]	DegreesTrue
DegreesTrue	DegreesTrue	none	(1..360)	INTEGER
departure	Cancel	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	CoordinateAccept	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	CoordinateInitial	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	CoordinateNegotiate	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	CoordinateReject	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	CoordinateStandby	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	CoordinateUpdate	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	GeneralPoint	SEQUENCE	[2] OPTIONAL	DepartureAirportTime
departure	Notify	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	SurveillanceGeneral	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferComm	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferCommAssume	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferConditionsAccept	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferConditionsProposal	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferControl	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferControlAssume	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferControlReject	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferInitiate	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
departure	TransferRequest	SEQUENCE	[1] OPTIONAL	DepartureAirportTime
descent	ReleaseIndicator	ENUMERATED	-1	
destination	Cancel	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	CoordinateAccept	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	CoordinateInitial	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	CoordinateNegotiate	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	CoordinateReject	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	CoordinateStandby	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	CoordinateUpdate	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	GeneralPoint	SEQUENCE	[3] OPTIONAL	DestinationAirport
destination	Notify	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	SurveillanceGeneral	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferComm	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferCommAssume	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferConditionsAccept	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferConditionsProposal	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferControl	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferControlAssume	SEQUENCE	[2] OPTIONAL	DestinationAirport

Identifier	In type	Constructor	Tag/Size	Of type
destination	TransferControlReject	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferInitiate	SEQUENCE	[2] OPTIONAL	DestinationAirport
destination	TransferRequest	SEQUENCE	[2] OPTIONAL	DestinationAirport
DestinationAirport	DestinationAirport	none		Airport
direction	VerticalChange	SEQUENCE	[0]	VerticalDirection
directRouting	ExecutiveData	SEQUENCE	[4] OPTIONAL	DirectRouting
distance	PlaceBearingDistance	SEQUENCE	[1]	Distance
distancekm	Distance	CHOICE	[1]	Distancekm
Distancekm	Distancekm	none	(0..2000)	INTEGER
distanceNM	Distance	CHOICE	[0]	DistanceNM
DistanceNM	DistanceNM	none	(0..1000)	INTEGER
down	VerticalDirection	ENUMERATED	-1	
east	LongitudeDirection	ENUMERATED	0	
edecca	ComNavEquipmentStatus	ENUMERATED	-3	
emergencyfreetext	InfoData	CHOICE	[4]	EmergencyFreeText
enddata	AIDC-crd-end-apdu	SEQUENCE	[0]	Enddata
errorCode	ApplicationErrorData	SEQUENCE	[2]	ErrorCode
errorData	ApplicationErrorData	SEQUENCE	[3] OPTIONAL	ErrorData
ErrorData	ErrorData	none	(SIZE(1.. 256))	BIT STRING
executivedata	GeneralExecutiveData	SEQUENCE	[2]	ExecutiveData
executiveData	TransferComm	SEQUENCE	[3] OPTIONAL	ExecutiveData
executiveData	TransferConditionsProposal	SEQUENCE	[3] OPTIONAL	ExecutiveData
executiveData	TransferControl	SEQUENCE	[3] OPTIONAL	ExecutiveData
executiveData	TransferInitiate	SEQUENCE	[3] OPTIONAL	ExecutiveData
fadf	ComNavEquipmentStatus	ENUMERATED	-4	
fix1	DirectRouting	SEQUENCE	[1]	Position
fix2	DirectRouting	SEQUENCE	[0] OPTIONAL	Position
FixName	FixName	none	(SIZE(1..5))	IA5String
fixName	PlaceBearing	SEQUENCE	[0]	FixName
fixName	Position	CHOICE	[0]	FixName
fixName	PublishedIdentifier	SEQUENCE	[0]	FixName
flightID	Cancel	SEQUENCE	[0]	FlightID
flightID	CoordinateAccept	SEQUENCE	[0]	FlightID
flightID	CoordinateInitial	SEQUENCE	[0]	FlightID
flightID	CoordinateNegotiate	SEQUENCE	[0]	FlightID
flightID	CoordinateReject	SEQUENCE	[0]	FlightID
flightID	CoordinateStandby	SEQUENCE	[0]	FlightID
flightID	CoordinateUpdate	SEQUENCE	[0]	FlightID
flightID	EmergencyFreeText	SEQUENCE	[1]	FlightID
flightID	GeneralExecutiveData	SEQUENCE	[0]	FlightID
flightID	GeneralFreeText	SEQUENCE	[1]	FlightID
flightID	GeneralPoint	SEQUENCE	[1]	FlightID
flightID	Notify	SEQUENCE	[0]	FlightID
flightID	SurveillanceGeneral	SEQUENCE	[0]	FlightID
flightID	TransferComm	SEQUENCE	[0]	FlightID
flightID	TransferCommAssume	SEQUENCE	[0]	FlightID
flightID	TransferConditionsAccept	SEQUENCE	[0]	FlightID

Identifier	In type	Constructor	Tag/Size	Of type
flightID	TransferConditionsProposal	SEQUENCE	[0]	FlightID
flightID	TransferControl	SEQUENCE	[0]	FlightID
flightID	TransferControlAssume	SEQUENCE	[0]	FlightID
flightID	TransferControlReject	SEQUENCE	[0]	FlightID
flightID	TransferInitiate	SEQUENCE	[0]	FlightID
flightID	TransferRequest	SEQUENCE	[0]	FlightID
flightRule	FlightRuleFlightType	SEQUENCE	[0]	FlightRule
flightRuleFlightType	CoordinateInitial	SEQUENCE	[3] OPTIONAL	FlightRuleFlightType
flightRuleFlightType	GeneralPoint	SEQUENCE	[4]	FlightRuleFlightType
flightRuleFlightType	Notify	SEQUENCE	[3] OPTIONAL	FlightRuleFlightType
flightType	FlightRuleFlightType	SEQUENCE	[1]	FlightType
freeText	EmergencyFreeText	SEQUENCE	[2]	FreeText
FreeText	FreeText	none	(SIZE (1..256))	IA5String
freeText	GeneralFreeText	SEQUENCE	[2]	FreeText
frequency	CoordinateAccept	SEQUENCE	[3] OPTIONAL	Frequency
frequency	GeneralExecutiveData	SEQUENCE	[1]	Frequency
frequency	TransferConditionsAccept	SEQUENCE	[3] OPTIONAL	Frequency
frequency	TransferRequest	SEQUENCE	[3] OPTIONAL	Frequency
frequencyHF	Frequency	CHOICE	[0]	FrequencyHF
FrequencyHF	FrequencyHF	none	(2850..28000)	INTEGER
frequencySatChannel	Frequency	CHOICE	[3]	FrequencySatChannel
FrequencySatChannel	Frequency	none	(SIZE(12))	NumericString
frequencyUHF	Frequency	CHOICE	[2]	FrequencyUHF
FrequencyUHF	Frequency	none	(9000..15999)	INTEGER
frequencyVHFChannel	Frequency	CHOICE	[1]	FrequencyVHFChannel
FrequencyVHFChannel	FrequencyVHFChannel	none	(23600..27398)	INTEGER
functionalAddress	EmergencyFreeText	SEQUENCE	[0] OPTIONAL	FunctionalAddress
functionalAddress	GeneralFreeText	SEQUENCE	[0] OPTIONAL	FunctionalAddress
functionalAddress	GeneralPoint	SEQUENCE	[0] OPTIONAL	FunctionalAddress
FunctionalAddress	FunctionalAddress	none	(SIZE(1..18))	IA5String
generalAviation	FlightType	ENUMERATED	-2	
generalexecutivedata	InfoData	CHOICE	[0]	GeneralExecutiveData
generalfreetext	InfoData	CHOICE	[3]	GeneralFreeText
generalpoint	InfoData	CHOICE	[1]	GeneralPoint
ggnss	ComNavEquipmentStatus	ENUMERATED	-5	
heading	ExecutiveData	SEQUENCE	[2] OPTIONAL	DegreesMagnetic
hf	DataLink	ENUMERATED	0	
hhfRtf	ComNavEquipmentStatus	ENUMERATED	-6	
high	WakeTurbulenceCategory	ENUMERATED	0	
hours	Time	SEQUENCE	[0]	TimeHours
hoursminute	Timehhmmss	SEQUENCE	Time	
ifr	FlightRule	ENUMERATED	0	
ifrfirst	FlightRule	ENUMERATED	-2	
iinertialNavigation	ComNavEquipmentStatus	ENUMERATED	-7	
infodata	AIDC-inf-tfr-apdu	SEQUENCE	[0]	InfoData
invalidAircraftIdentification	ErrorCode	ENUMERATED	-28	
invalidAircraftType	ErrorCode	ENUMERATED	-1	

Identifier	In type	Constructor	Tag/Size	Of type
invalidAirframeID	ErrorCode	ENUMERATED	-31	
invalidATSRouteDesignator	ErrorCode	ENUMERATED	-42	
invalidATWLevelTolerance	ErrorCode	ENUMERATED	-14	
invalidBeaconCodeOctalDigit	ErrorCode	ENUMERATED	-3	
invalidcalledICAOFacilityDesignation	ErrorCode	ENUMERATED	-253	
invalidcallingICAOFacilityDesignation	ErrorCode	ENUMERATED	-252	
invalidComNavEquipmentStatus	ErrorCode	ENUMERATED	-15	
invalidDataLink	ErrorCode	ENUMERATED	-17	
invalidDistanceKm	ErrorCode	ENUMERATED	-40	
invalidDistanceNM	ErrorCode	ENUMERATED	-41	
invalidFixName	ErrorCode	ENUMERATED	-4	
invalidFlightRule	ErrorCode	ENUMERATED	-32	
invalidFlightType	ErrorCode	ENUMERATED	-33	
invalidFrequencyHF	ErrorCode	ENUMERATED	-34	
invalidFrequencySatChannel	ErrorCode	ENUMERATED	-37	
invalidFrequencyUHF	ErrorCode	ENUMERATED	-36	
invalidFrequencyVHFChannel	ErrorCode	ENUMERATED	-35	
invalidFunctionalAddress	ErrorCode	ENUMERATED	-38	
invalidLatitude	ErrorCode	ENUMERATED	-7	
invalidLevelFeet	ErrorCode	ENUMERATED	-10	
invalidLevelFlightLevel	ErrorCode	ENUMERATED	-12	
invalidLevelFlightLevelMetric	ErrorCode	ENUMERATED	-13	
invalidLevelMeters	ErrorCode	ENUMERATED	-11	
invalidLongitude	ErrorCode	ENUMERATED	-8	
invalidmsgnumber	ErrorCode	ENUMERATED	-250	
invalidNavaid	ErrorCode	ENUMERATED	-5	
invalidNumberOfAircraft	ErrorCode	ENUMERATED	0	
invalidreferenceid	ErrorCode	ENUMERATED	-251	
invalidRegistration	ErrorCode	ENUMERATED	-30	
invalidReleaseIndicator	ErrorCode	ENUMERATED	-39	
invalidSelcal	ErrorCode	ENUMERATED	-29	
invalidSpeedGround	ErrorCode	ENUMERATED	-18	
invalidSpeedGroundMetric	ErrorCode	ENUMERATED	-19	
invalidSpeedIndicated	ErrorCode	ENUMERATED	-21	
invalidSpeedIndicatedMetric	ErrorCode	ENUMERATED	-22	
invalidSpeedMach	ErrorCode	ENUMERATED	-20	
invalidSpeedTrue	ErrorCode	ENUMERATED	-23	
invalidSpeedTrueMetric	ErrorCode	ENUMERATED	-24	
invalidSSREquipmentAvailable	ErrorCode	ENUMERATED	-16	
invalidTime	ErrorCode	ENUMERATED	-9	
invalidtimestamp	ErrorCode	ENUMERATED	-254	
invalidTrackName	ErrorCode	ENUMERATED	-43	
invalidVerticalDirection	ErrorCode	ENUMERATED	-25	
invalidVerticalRateEnglish	ErrorCode	ENUMERATED	-26	
invalidVerticalRateMetric	ErrorCode	ENUMERATED	-27	
invalidWakeTurbulenceCategory	ErrorCode	ENUMERATED	-2	
itransponderModeSID	SSREquipmentAvailable	ENUMERATED	-5	

Identifier	In type	Constructor	Tag/Size	Of type
latitude	LatitudeLongitude	SEQUENCE	[0]	Latitude
latitudeDegrees	Latitude	SEQUENCE	[0]	LatitudeDegrees
LatitudeDegrees	LatitudeDegrees	none	(0..90000)	INTEGER
latitudeDirection	Latitude	SEQUENCE	[3]	LatitudeDirection
latitudeLongitude	PlaceBearing	SEQUENCE	[1] OPTIONAL	LatitudeLongitude
latitudeLongitude	Position	CHOICE	[3]	LatitudeLongitude
latitudeLongitude	PublishedIdentifier	SEQUENCE	[1]	LatitudeLongitude
latitudeLongitude	RouteInformation	CHOICE	[1]	LatitudeLongitude
latitudeLongitude	TrackDetail	SEQUENCE	[1]	LatitudeLongitude
latitudeMinutes	Latitude	SEQUENCE	[1] OPTIONAL	DegreeMinutes
latitudeSeconds	Latitude	SEQUENCE	[2] OPTIONAL	DegreeSeconds
level	ATWLevel	SEQUENCE	[1]	Level
level	ExecutiveData	SEQUENCE	[1] OPTIONAL	Level
level	Route	SEQUENCE	[2]	Level
level	TrackData	SEQUENCE	[2]	Level
levelFeet	Level	CHOICE	[0]	LevelFeet
LevelFeet	LevelFeet	none	(-60.. 7000)	INTEGER
levelFlightLevel	Level	CHOICE	[2]	LevelFlightLevel
LevelFlightLevel	LevelFlightLevel	none	(30..700)	INTEGER
levelFlightLevelMetric	Level	CHOICE	[3]	LevelFlightLevelMetric
LevelFlightLevelMetric	LevelFlightLevelMetric	none	(100..2500)	INTEGER
levelMetre	Level	CHOICE	[1]	LevelMetre
LevelMetre	LevelMetre	none	(-30..25000)	INTEGER
lils	ComNavEquipmentStatus	ENUMERATED	-8	
longitude	LatitudeLongitude	SEQUENCE	[1]	Longitude
longitudeDegrees	Longitude	SEQUENCE	[0]	LongitudeDegrees
LongitudeDegrees	LongitudeDegrees	none	(0..180000)	INTEGER
longitudeDirection	Longitude	SEQUENCE	[3]	LongitudeDirection
longitudeMinutes	Longitude	SEQUENCE	[1] OPTIONAL	DegreeMinutes
longitudeSeconds	Longitude	SEQUENCE	[2] OPTIONAL	DegreeSeconds
low	WakeTurbulenceCategory	ENUMERATED	-2	
medium	WakeTurbulenceCategory	ENUMERATED	-1	
MessageNumber	MessageNumber	none	(0..999999)	INTEGER
messageType	ApplicationErrorData	SEQUENCE	[0]	MessageType
military	FlightType	ENUMERATED	-3	
minutes	Time	SEQUENCE	[1]	TimeMinutes
modeS	DataLink	ENUMERATED	-1	
momega	ComNavEquipmentStatus	ENUMERATED	-9	
month	Date	SEQUENCE	[1]	Month
Month	Month	none	(1..12)	INTEGER
msgnumber	AIDC-crd-end-apdu	SEQUENCE	[2]	MessageNumber
msgnumber	AIDC-crd-ngtt-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-crd-start-apdu	SEQUENCE	[3]	MessageNumber
msgnumber	AIDC-crd-stndby-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-end-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-inf-tfr-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-nfy-apdu	SEQUENCE	[3]	MessageNumber

Identifier	In type	Constructor	Tag/Size	Of type
msgnumber	AIDC-tfr-accept-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-tfr-cntrl-Req-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-tfr-cntrl-Rsp-apdu	SEQUENCE	[2]	MessageNumber
msgnumber	AIDC-tfr-comm-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-tfr-comm-assm-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-tfr-init-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-tfr-prpsl-apdu	SEQUENCE	[1]	MessageNumber
msgnumber	AIDC-tfr-rqst-apdu	SEQUENCE	[1]	MessageNumber
Navaid	Navaid	none	(SIZE(1..4))	IA5String
navaid	Position	CHOICE	[1]	Navaid
nnil	SSREquipmentAvailable	ENUMERATED	0	
nonScheduledAirTransport	FlightType	ENUMERATED	-1	
north	LatitudeDirection	ENUMERATED	0	
notify	AIDC-nfy-apdu	SEQUENCE	[2]	Notify
numberOfAircraft	AircraftNumberType	SEQUENCE	[0] OPTIONAL	NumberOfAircraft
NumberOfAircraft	NumberOfAircraft	none	(1..2)	INTEGER
otherFlights	FlightType	ENUMERATED	-4	
otherinfo	Cancel	SEQUENCE	[4] OPTIONAL	OtherInformation
otherInfo	CoordinateInitial	SEQUENCE	[9] OPTIONAL	OtherInformation
otherInfo	GeneralPoint	SEQUENCE	[10] OPTIONAL	OtherInformation
otherInfo	Notify	SEQUENCE	[9] OPTIONAL	OtherInformation
OtherInformation	OtherInformation	none		FreeText
ovor	ComNavEquipmentStatus	ENUMERATED	-10	
pdoppler	ComNavEquipmentStatus	ENUMERATED	-11	
placeBearing	PlaceBearingDistance	SEQUENCE	[0]	PlaceBearing
placeBearingDistance	Position	CHOICE	[4]	PlaceBearingDistance
placeBearingDistance	RouteInformation	CHOICE	[3]	PlaceBearingDistance
PlaceBearingPlaceBearing	PlaceBearingPlaceBearing	SEQUENCE OF	SIZE (2)	PlaceBearing
placeBearingPlaceBearing	RouteInformation	CHOICE	[2]	PlaceBearingPlaceBearing
position	Route	SEQUENCE	[0]	Position
position	TrackData	SEQUENCE	[0]	Position
protocolerror	ProviderAbortReason	ENUMERATED	0	
providererror	ProviderAbortReason	ENUMERATED	-3	
ptransponderModeSPA	SSREquipmentAvailable	ENUMERATED	-4	
publishedIdentifier	RouteInformation	CHOICE	[0]	PublishedIdentifier
rate	VerticalChange	SEQUENCE	[1]	VerticalRate
reason	AIDC-ucf-apdu	SEQUENCE	[1] OPTIONAL	ApplicationErrorData
referenceid	AIDC-ucf-apdu	SEQUENCE	[2]	MessageNumber
registration	FlightID	SEQUENCE	[2] OPTIONAL	Registration
Registration	Registration	none	(SIZE(5))	IA5String
rejected	Result	ENUMERATED	-1	
rejectedpermanent	ProviderAbortReason	ENUMERATED	-4	
rejectedtransient	ProviderAbortReason	ENUMERATED	-5	
releaseIndicator	TransferComm	SEQUENCE	[4] OPTIONAL	ReleaseIndicator
result	AIDC-crd-end-apdu	SEQUENCE	[1]	Result
result	AIDC-tfr-cntrl-Rsp-apdu	SEQUENCE	[1]	Result
result	AIDC-ucf-apdu	SEQUENCE	[0]	Result

Identifier	In type	Constructor	Tag/Size	Of type
route	CoordinateInitial	SEQUENCE	[8] OPTIONAL	Route
route	CoordinateNegotiate	SEQUENCE	[4] OPTIONAL	Route
route	CoordinateUpdate	SEQUENCE	[5] OPTIONAL	Route
route	GeneralPoint	SEQUENCE	[9] OPTIONAL	Route
route	Notify	SEQUENCE	[8] OPTIONAL	Route
rrnavRouteEquipment	ComNavEquipmentStatus	ENUMERATED	-12	
satcom	DataLink	ENUMERATED	-2	
satransponderModeSPAID	SSREquipmentAvailable	ENUMERATED	-6	
scheduledAirTransport	FlightType	ENUMERATED	0	
seconds	Timehhmmss	SEQUENCE	TimeSeconds	
selcal	FlightID	SEQUENCE	[1] OPTIONAL	Selcal
Selcal	Selcal	none	(SIZE(4))	IA5String
south	LatitudeDirection	ENUMERATED	-1	
speed	ExecutiveData	SEQUENCE	[0] OPTIONAL	Speed
speedGround	Route	SEQUENCE	[3]	SpeedGround
speedGround	Speed	CHOICE	[0]	SpeedGround
SpeedGround	SpeedGround	none	(-50..2000)	INTEGER
speedGround	TrackData	SEQUENCE	[3]	SpeedGround
speedGroundMetric	Speed	CHOICE	[1]	SpeedGroundMetric
SpeedGroundMetric	SpeedGroundMetric	none	(-100..4000)	INTEGER
speedIndicated	Speed	CHOICE	[3]	SpeedIndicated
SpeedIndicated	SpeedIndicated	none	(0..400)	INTEGER
speedIndicatedMetric	Speed	CHOICE	[4]	SpeedIndicatedMetric
SpeedIndicatedMetric	SpeedIndicatedMetric	none	(0..800)	INTEGER
speedMach	Speed	CHOICE	[2]	SpeedMach
SpeedMach	SpeedMach	none	(500..4000)	INTEGER
speedTrue	Speed	CHOICE	[5]	SpeedTrue
SpeedTrue	SpeedTrue	none	(0..2000)	INTEGER
speedTrueMetric	Speed	CHOICE	[6]	SpeedTrueMetric
SpeedTrueMetric	SpeedTrueMetric	none	(0..4000)	INTEGER
ssrEquipmentAvailable	CNSEquipment	SEQUENCE	[1]	SSREquipmentAvailable
startdata	AIDC-crd-start-apdu	SEQUENCE	[2]	Startdata
surveillancegeneral	InfoData	CHOICE	[2]	SurveillanceGeneral
time	DepartureAirportTime	SEQUENCE	[1] OPTIONAL	Time
time	Route	SEQUENCE	[1]	Time
time	TrackData	SEQUENCE	[1]	Time
timehhmmss	YMDHMS	SEQUENCE	[1]	Timehhmmss
TimeHours	TimeHours	none	(0..23)	INTEGER
TimeMinutes	TimeMinutes	none	(0..59)	INTEGER
timerexpired	ProviderAbortReason	ENUMERATED	-1	
TimeSeconds	TimeSeconds	none	(0..59)	INTEGER
timestamp	Cancel	SEQUENCE	[5]	YMDHMS
timestamp	CoordinateAccept	SEQUENCE	[4]	YMDHMS
timestamp	CoordinateInitial	SEQUENCE	[10]	YMDHMS
timestamp	CoordinateNegotiate	SEQUENCE	[5]	YMDHMS
timestamp	CoordinateReject	SEQUENCE	[3]	YMDHMS
timestamp	CoordinateStandby	SEQUENCE	[3]	YMDHMS

Identifier	In type	Constructor	Tag/Size	Of type
timestamp	CoordinateUpdate	SEQUENCE	[6]	YMDHMS
timestamp	EmergencyFreeText	SEQUENCE	[3]	YMDHMS
timestamp	GeneralExecutiveData	SEQUENCE	[4]	YMDHMS
timestamp	GeneralFreeText	SEQUENCE	[3]	YMDHMS
timestamp	GeneralPoint	SEQUENCE	[11]	YMDHMS
timestamp	Notify	SEQUENCE	[10]	YMDHMS
timestamp	SurveillanceGeneral	SEQUENCE	[4]	YMDHMS
timestamp	TransferComm	SEQUENCE	[5]	YMDHMS
timestamp	TransferCommAssume	SEQUENCE	[3]	YMDHMS
timestamp	TransferConditionsAccept	SEQUENCE	[4]	YMDHMS
timestamp	TransferConditionsProposal	SEQUENCE	[4]	YMDHMS
timestamp	TransferControl	SEQUENCE	[4]	YMDHMS
timestamp	TransferControlAssume	SEQUENCE	[3]	YMDHMS
timestamp	TransferControlReject	SEQUENCE	[3]	YMDHMS
timestamp	TransferInitiate	SEQUENCE	[5]	YMDHMS
timestamp	TransferRequest	SEQUENCE	[4]	YMDHMS
trackData	SurveillanceGeneral	SEQUENCE	[3]	TrackData
trackData	TransferInitiate	SEQUENCE	[4] OPTIONAL	TrackData
trackDetail	RouteInformation	CHOICE	[5]	TrackDetail
trackName	TrackDetail	SEQUENCE	[0]	TrackName
TrackName	TrackName	none	(SIZE(1..6))	IA5String
transfercomm	AIDC-tfr-comm-apdu	SEQUENCE	[0]	TransferComm
transfercommassume	AIDC-tfr-comm-assm-apdu	SEQUENCE	[0]	TransferCommAssume
transferconditionsaccept	AIDC-tfr-accept-apdu	SEQUENCE	[0]	TransferConditionsAccept
transferconditionsproposal	AIDC-tfr-prpsl-apdu	SEQUENCE	[0]	TransferConditionsProposal
transfercontrol	AIDC-tfr-cntrl-Req-apdu	SEQUENCE	[0]	TransferControl
transfercontrolassume	TransferControlData	CHOICE	[0]	TransferControlAssume
transfercontroldata	AIDC-tfr-cntrl-Rsp-apdu	SEQUENCE	[0]	TransferControlData
transfercontrolreject	TransferControlData	CHOICE	[1]	TransferControlReject
transferinitiate	AIDC-tfr-init-apdu	SEQUENCE	[0]	TransferInitiate
transferrequest	AIDC-tfr-rqst-apdu	SEQUENCE	[0]	TransferRequest
trueTrackAngle	Route	SEQUENCE	[4]	TrueTrackAngle
trueTrackAngle	TrackData	SEQUENCE	[4]	TrueTrackAngle
TrueTrackAngle	TrueTrackAngle	none		Degrees
ttacan	ComNavEquipmentStatus	ENUMERATED	-13	
turns	ReleaseIndicator	ENUMERATED	-3	
undefinederror	ProviderAbortReason	ENUMERATED	-2	
unknown	ErrorCode	ENUMERATED	-255	
up	VerticalDirection	ENUMERATED	0	
uuhfRTF	ComNavEquipmentStatus	ENUMERATED	-14	
verticalRateEnglish	VerticalRate	CHOICE	[0]	VerticalRateEnglish
VerticalRateEnglish	VerticalRateEnglish	none	(0..3000)	INTEGER
verticalRateMetric	VerticalRate	CHOICE	[1]	VerticalRateMetric
VerticalRateMetric	VerticalRateMetric	none	(0..1000)	INTEGER
vertRate	ExecutiveData	SEQUENCE	[3] OPTIONAL	VerticalChange
vfr	FlightRule	ENUMERATED	-1	
vfrfirst	FlightRule	ENUMERATED	-3	

Identifier	In type	Constructor	Tag/Size	Of type
vhf	DataLink	ENUMERATED	-3	
vvhfRTF	ComNavEquipmentStatus	ENUMERATED	-15	
wakeTurbulenceCategory	AircraftNumberType	SEQUENCE	[2] OPTIONAL	WakeTurbulenceCategory
west	LongitudeDirection	ENUMERATED	-1	
xatransponderModeS	SSREquipmentAvailable	ENUMERATED	-3	
year	Date	SEQUENCE	[0]	Year
Year	Year	none	(1996..2095)	INTEGER

7.7 Example Scenarios

7.7.1 Transfer communications procedure

7.7.1.1 This section considers two simple coordination and transfer examples, showing the use of AIDC in combination with the ATN CPDLC air-ground application. These scenarios show the use of the AIDC transfer of communications procedure, used when the FIR boundary is under full surveillance.

7.7.1.2 CPDLC airborne and ground systems, and supporting procedures, will ensure that Transfer of Data Authority using the CPDLC application messages can be carried out in the following circumstances:

a) independent of the transferring and receiving ATSUs ground/ground data communication capability;

b) when both the transferring and receiving ATSUs are equipped for air/ground data link; and

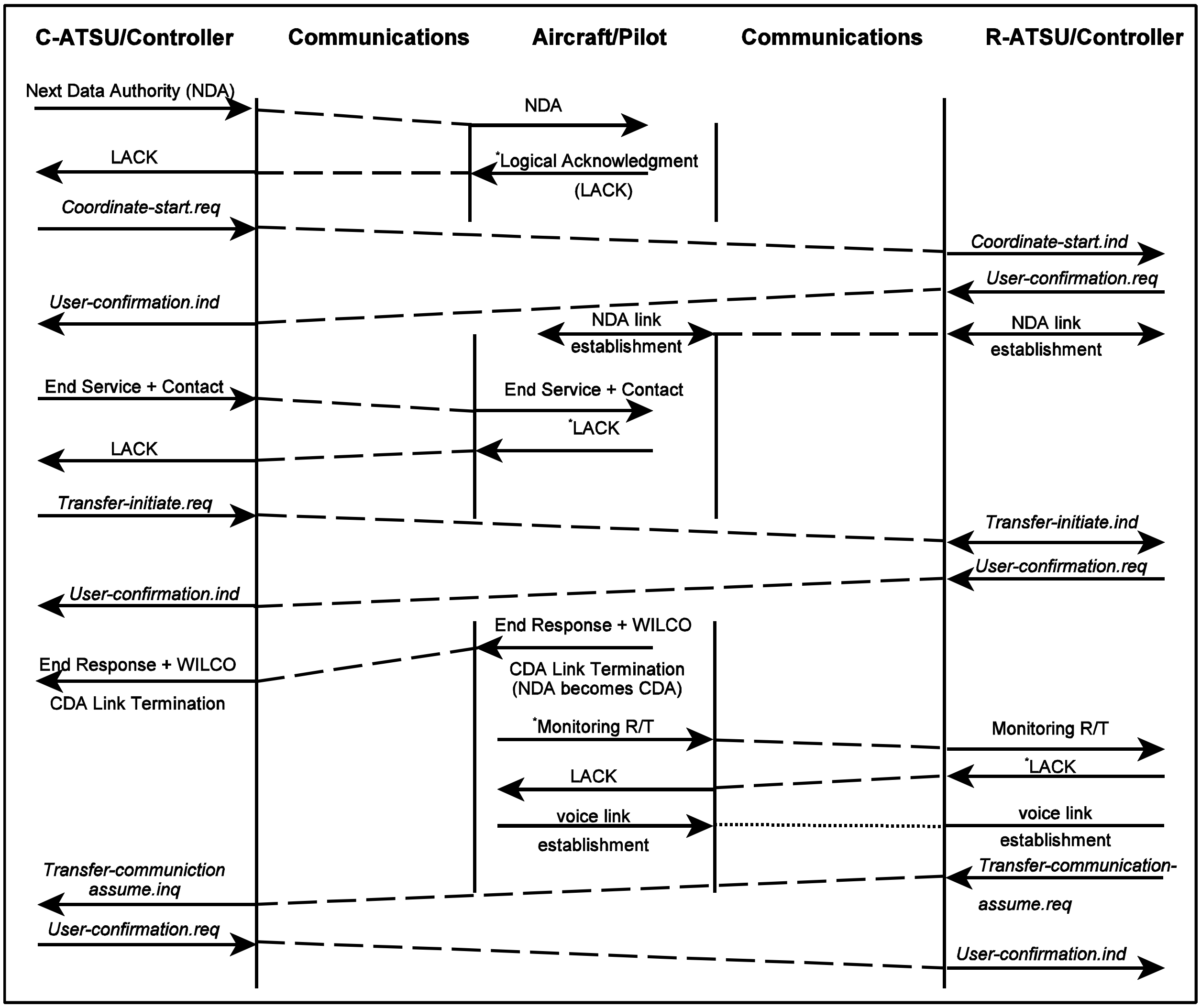
c) when only the transferring ATSU is equipped for air/ground data link.

7.7.1.2.1 ATSUs may elect to use ground/ground data exchanges in support of the transfer of data authority, subject to bilateral agreements, local procedures and local infrastructure. Such ground/ground exchanges are not required for successful completion of the transfer.

7.7.1.2.2 In these scenarios ground/ground data communication in support of ATC transfer of data authority is provided by AIDC. In practice the AIDC SARPs would be complemented by regional supplementary material and bilateral agreements.

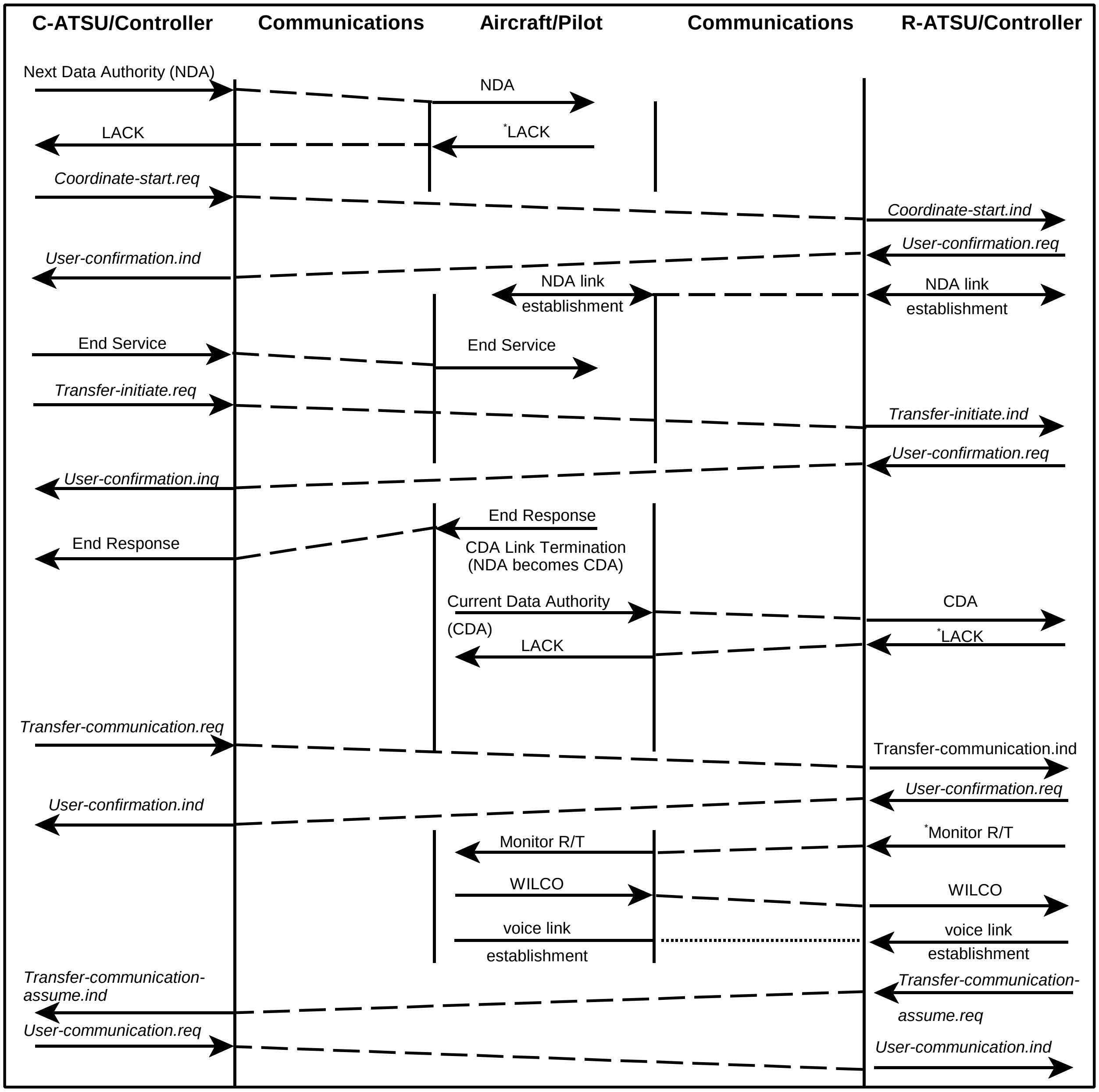
7.7.1.2.3 Figures 7.7-1 and 7.7-2 illustrate transfer of data and communications authority with supporting ground/ground connectivity. In these figures we show the CPDLC exchanges in terms of operational messages since the detailed protocol exchanges are not relevant. The AIDC exchanges are shown in terms of primitives of the AIDC abstract service.

7.7.1.2.4 The figures show two possible sequences of communications exchanges. The AIDC and CPDLC specifications support other sequences of exchanges to achieve the same operational result.



Key:
AIDC primitive
CPDLC message
* indicates an optional message

Figure 7.7-1. Example of Transfer of Data and Communications Authority (C-ATSU relinquishes responsibility)



Key:
AIDC primitive
CPDLC message
* indicates an optional message

Figure 7.7-2. Example of Transfer of Communications Authority (R-ATSU assumes responsibility)

7.7.2 Transfer control procedure

7.7.2.1 This section shows the use of the AIDC transfer of control procedure, used when the FIR boundary is not under full surveillance, or otherwise by mutual agreement.

7.7.2.2 In Figure 7.7-3 the C-ATSU proposes coordination conditions using the Coordinate-Start primitive. The R-ATSU responds by offering alternative conditions using Coordinate-Negotiate, which are accepted by the C-ATSU. The flight is transferred using the Transfer-control primitive.

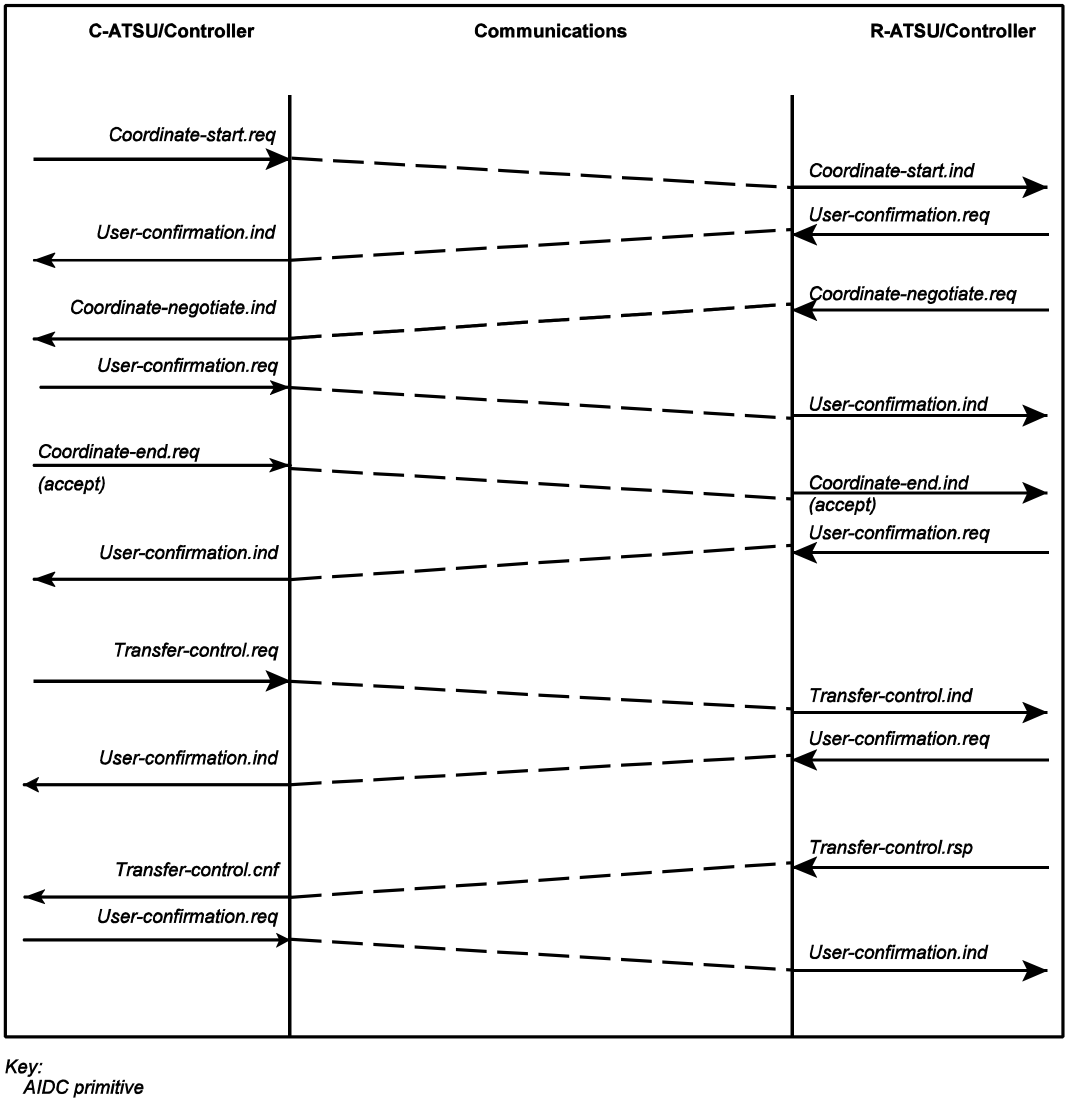


Figure 7.7-3. Example of Transfer of Control Procedure

7.7.2.3 In practice the Coordinate-Standby primitive may be used as a response to one or both of Coordinate-Start and Coordinate-Negotiate, followed by the response shown. This would be necessary if the processing of either primitive might otherwise cause timer t_{1R} to expire; for instance, if manual intervention by the controller is needed.

7.8 Example Encoding

7.8.1 AIDC-ucf PDU

7.8.1.1 The following is an example of the encoding of a hypothetical AIDC-ucf PDU.

Table 7.8-1. AIDC-ucf PDU encoding

Element	Sub-element	Value	Encoding	Comments
AIDC-APDU CHOICE	extension bit	false	B‘0’	no extension present
aidc-ucf-apdu [0]	CHOICE tag	0	B‘0000’	tags are 0..15
SEQUENCE	options bitmap		B‘0’	first of one option absent
result [0]	SEQUENCE tag	0	none	
ENUMERATED		accepted (0)	B‘0’	treat as INTEGER (0..1)
reason [1] OPTIONAL			none	No encoding as not present
referenceid [2]	INTEGER (0..999999)	666	B‘00000000001010011010’	999999 needs 20 bits

7.8.1.2 Thus, the PDU is encoded as B‘0 0000 0 0 00000000001010011010’ or in hexadecimal as H‘000053’ followed by a trailing B‘010’.

7.8.1.3 As illustrated in Part IV.7.6, this encoding is then embedded as the item presentation-data-values in a SEQUENCE OF PDV-list and the resulting bit string is padded to a multiple of eight bits for transmission in the User Data field of a T-DATA PDU (there being no session or presentation protocol control information since the null encoding option is selected).