CPDLC Application Design Document

Deliverable ID:	**D4.2**
Project Acronym:	**ATMACA**
Grant:	**101167070**
Call:	**HORIZON-SESAR-2023-DES-ER-02**
Topic:	**HORIZON-SESAR-2023-DES-ER2-WA2-2**
Consortium Coordinator:	**Eskişehir Teknik Üniversitesi**
Edition date:	**14 August 2025**
Edition:	**01.03**
Status:	**Final**
Classification:	**PU**

Abstract

This document outlines the design of the Controller–Pilot DataLink Communications (CPDLC) application within the ATMACA (Air Traffic Management and Communication Over ATN/IPS) framework, developed under the SESAR 3 (Single European Sky ATM Research) program. The application enables digital communication between pilots and air traffic controllers using International Civil Aviation Organization (ICAO) -standardised messages and operates over an Internet Protocol (IP)-based Aeronautical Telecommunication Network / Internet Protocol Suite (ATN/IPS) network. It integrates session continuity, context management, and role-based access through Datalink Context Management (DLCM) middleware, supporting services such as Data Link Initiation Capability (DLIC), Air Traffic Control Communications Management (ACM), Air Traffic Control Clearance (ACL), and Downstream Clearance Service (DSC). With a modular architecture to be used with the user-friendly Human–Machine Interface (HMI), and compatibility with System Wide Information Management (SWIM), the solution enhances interoperability, automation, and operational flexibility in modern air traffic management systems.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Authoring & Approval

Author(s) of the document

	Date
Rosa María Arnaldo Valdés / UPM	08/07/2025
Raquel Delgado-Aguilera Jurado / UPM	08/07/2025
Guillermo Martín Vicente / UPM	08/07/2025
Pedro Reinaldos Manzanares / UPM	08/07/2025

Reviewed by

	Date
Hassna Louadah / DMU	23/07/2025
Raouf Hamzaoui / DMU	23/07/2025
Feng Chen / DMU	23/07/2025
Cem Çetek / ESTU	23/07/2025
Fulya Aybek Çetek / ESTU	23/07/2025

Approved for submission to the SESAR 3 JU by[1]

Name / Organisation	Date
Fulya Aybek Çetek / ESTU	14/08/2025
Stefano Bonelli and Tommaso Vendruscolo / DBL	14/08/2025
Raouf Hamzaoui / DMU	14/08/2025
Georges Mykoniatis / ENAC	14/08/2025
Ángel Ernesto Carmona Fernández / SAERCO	14/08/2025
Sergun Özmen / THY	14/08/2025
Rosa María Arnaldo Valdes / UPM	14/08/2025

Rejected by[2]

Company Name	Date

[1] Representatives of all the beneficiaries involved in the project
[2] Representatives of the beneficiaries involved in the project

Document History

Edition	Date	Status	Company Author	Justification
01.01	08/07/2025	Draft	UPM	Draft version
01.02	31/07/2025	Reviewed Version	UPM	Revision made in line with the reviewers' comments.
01.03	14/08/2025	Final Revised	UPM	Approval of the Consortium.

Copyright Statement

Disclaimer

ATMACA

AIR TRAFFIC MANAGEMENT AND COMMUNICATION OVER ATN/IPS (ATMACA)

ATMACA

This document is part of a project that has received funding from the SESAR 3 Joint Undertaking under grant agreement No 101167070 under European Union's Horizon Europe research and innovation programme.

The UK participant (De Montfort University) is supported by UKRI grant number 10121610.

Table of Contents

Abstract 1

1 INTRODUCTION 11

1.1 Purpose 11

1.2 Scope 11

1.3 Background 11

1.3.1 Mobility Management within ATMACA 11

1.3.2 The ATMACA Framework 12

1.4 Structure of the Document 17

2 OVERVIEW OF THE ATMACA FRAMEWORK 19

2.1 ATMACA Network Architecture 19

2.2 Protocol Description 20

2.2.1 Software Infrastructure 20

2.2.2 ATMACA Protocol Message Format 21

2.2.3 ATMACA Message Grammar Augmented Backus–Naur Form (ABNF) 22

2.2.4 ATMACA Messaging Concept 23

2.2.5 ATMACA Message Processing 24

3 CPDLC APPLICATION ARCHITECTURE 25

3.1 Architecture Overview 25

3.2 CPDLC's Role and Purpose 25

3.3 CPDLC's Main Functional Requirements 26

3.4 CPDLC Architecture and Framework Interactions 27

3.5 Functional Subsystem 29

3.5.1 Group 1 – Standard Datalink Services 31

3.5.2 Group 2 – Enhanced Ground Services 33

3.6 Datalink Services Workflow 36

3.6.1 Predeparture Clearance (DCL) 36

3.7 CPDLC Interactions within ATMACA 52

3.7.1 Interactions with ATMACA protocol Stack and Stack Interface 52

3.7.2 Interactions with DLCM 54

3.7.3 Interactions with HMI 63

4 CPDLC MESSAGE DEFINITION 66

4.1 ATMACA Message Format and Structure 66

4.1.1 ATMACA Message Structure 66

4.1.2 ATMACA DIX Structure 67

4.1.3 ABNF Grammer 68

4.1.4 Common DIXs 69

4.2 CPDLC Message Definition 70

4.2.1 CPDLC Message Format 74

4.2.2 ACM Messages Format 76

4.2.3 CPDLC Message List 77
5 APPLICATION REQUIREMENTS 79
5.1 Message Exchange and Communication Handling 79
5.2 Protocol Adaptation 79
5.3 Safety, Security and Compliance Features 79
5.4 Interoperability and Systems Integration 80
5.5 Performance and Compliance Requirements 80
6 USER GUIDE FOR INTEGRATION AND TESTING 81
6.1 CPDLC Application Testing 81
6.2 CPDLC Testing within the ATMACA Framework 81
6.3 CPDLC User Guide 81
6.3.1 Message Building 81
6.3.2 Application Initialization 82
6.3.3 Message Interchange 83
7 Conclusion 85
8 REFERENCES 87

List of Figures

Figure 1.1: ATMACA Framework.. 13

Figure 1.2: Key operational improvements aimed by ATMACA 13

Figure 1.3: Manual Handling of datalink logon/off processes 14

Figure 1.4: Automated Handling of datalink logon/off in ATMACA.. 15

Figure 1.5: Handover and Transfer of flights in ATMACA 16

Figure 1.6: Structured Attributes of the Flight Session.. 17

Figure 2.1: ATMACA's Network Architecture... 20

Figure 2.2. ATMACA's software architecture 21

Figure 2.3: ATMACA message structure 22

Figure 2.4: ATMACA ABNF 23

Figure 3.1.CPDLC Service architecture... 25

Figure 3.2: CPDLC's Module architecture 28

Figure 3.3: CPDLC's framework interactions.. 29

Figure 3.4: CPDLC Core Module - Class - Functional Architecture Level 30

Figure 3.5. DCL Service - Class Level Architecture 31

Figure 3.6. ACM Service - Class - Level Architecture.. 32

Figure 3.7.ACL Service - Class-Level Architecture 32

Figure 3.8. DSC Service - Class-Level Architecture 33

Figure 3.9: D-ATIS Service - Class-Level Architecture.. 34

Figure 3.10: D-TAXI Service - Class-Level Architecture 35

Figure 3.11: D-PUSHBACK Service - Class-Level Architecture 35

Figure 3.12: D-STARTUP Service - Class-Level Architecture 36

Figure 3.13: Departure Clearance Workflow - Strict Pair Mode (UDP).. 38

Figure 3.14: Departure Clearance Workflow - Hybrid Mode (TCP)... 39

Figure 3.15: D-Startup Clearance Workflow - Strict Pair Mode (UDP).. 40

Figure 3.16: Departure Clearance Workflow - Hybrid Mode (TCP)... 41

Figure 3.17: D-Pushback Workflow - Hybrid Mode (TCP) 42

Figure 3.18: D-Pushback Clearance Workflow - Strict Pair Mode (UDP) 43

Figure 3.19: D-Taxi Clearance Workflow - Strict Pair Mode (UDP) 44

Figure 3.20: D-Taxi Clearance Workflow - Hybrid Mode (TCP) 45

Figure 3.21: Transfer of Data Authority – T-ATSU Initiated (Strict Mode – UDP) 46

Figure 3.22: Transfer of Data Authority – T-ATSU Initiated (Hybrid Mode – TCP) 47

Figure 3.23: Uplink – RESUME NORMAL SPEED (Strict Mode – UDP) 48

Figure 3.24: Uplink – RESUME NORMAL SPEED (Hybrid Mode – TCP) 48

Figure 3.25: Downlink – REQUEST VMC DESCENT (Hybrid Mode – TCP) 49

Figure 3.26: Downlink – REQUEST VMC DESCENT (Strict Pair – UDP) 50

Figure 3.27: Downstream Clearance Workflow (UDP - Strict Pair Mode) 51

Figure 3.28: Downstream Clearance Workflow (TCP - Hybrid Mode) 52

Figure 3.29: ATMACA Messaging and Peer Management Architecture 53

Figure 3.30: ATMACA Routing Management Hierarchy 53

Figure 3.31: DLCM Module Interaction 54

Figure 3.32: Context Management Architectural Elements Diagram 55

Figure 3.33: Session Management Architectural Elements Diagram 57

Figure 3.34: Connection Management Architectural Elements Diagram 59

Figure 3.35: DLIC Functional Subsystem 61

Figure 3.36: DLIC Message Flow 61

Figure 3.37: Mobility Management Architectural Elements Diagram 63

Figure 3.38: Example of a CPDLC ATM Flight Deck HMI 64

Figure 4.1: ATMACA Message format 66

Figure 4.2: ATMACA Message Structure 66

Figure 4.3: ATMACA DIX Structure 67

Figure 4.4: ATMACA ABNF 69

Figure 4.5: ATMACA Request Message Format 70

Figure 4.6: ATMACA Response Message Format 70

Figure 4.7: Dictionary Message Addition Example 71

Figure 4.8: ACM Message List Example 72

Figure 4.9: AMC Message List Example 73

Figure 4.10: ACL Message List Example 74

Figure 4.11: ACM Message Format 77

List of Tables

Table 3.1: CPDLC Module - Linkage Table 30

Table 3.2: Context Main Structure 56

Table 3.3: Session Data Structure 58

Table 3.4: Peer Table Structure 60

Table 3.5: Types of Mobility 62

Table 4.1: CPDLC Urgency Attribute Types 75

Table 4.2: Alert Attribute Types 75

Table 4.3: Response Attribute Types (Uplink) 75

Table 4.4: Response Attribute types (Downlink) 76

Table 4.5: CPDLC Message List 77

List of Acronyms

Acronym	Name
ACM	ATC Communication Management
ACL	ATC Clearance
API	Application Programming Interface
ATC	Air Traffic Control
ATCo	Air Traffic Controller
ATM	Air Traffic Management
ATMACA	Air Traffic Management and Communication over ATN/IPS
ATN	Aeronautical Telecommunication Network
ATS	Air Traffic Services
ATSU	Air Traffic Services Unit
CPDLC	Controller-Pilot DataLink Communications
DCL	Departure Clearance
DFIS	Digital Flight Information Services
DIX	Datalink Information eXchange
DLIC	DataLink Initiation Capability
DLCM	DataLink Context Management
DSC	Downstream Clearance Service
EUROCONTROL	European Organization for the Safety of Air Navigation
FDPS	Flight Data Processing System
FCDI	Future Connectivity and Digital Infrastructure
FIR	Flight Information Region
FL	Flight Level
GB-LISP	Ground-based Locator/ID Separation Protocol
GOLD	Global Operational Datalink
GRO	Green Route Operations
HMI	Human Machine Interface
IA5	International Alphabet No. 5
ICAO	International Civil Aviation Organization
IP	Internet Protocol
IPS	Internet Protocol Suite
KPI	Key Performance Indicator

LACK	Logical Acknowledgement
OSI	Open Systems Interconnection
PMIPv6	Proxy Mobile IP version 6
QoS	Quality of Service
RBAC	Role-Based Access Control
SATCOM	Satellite Communications
SESAR	Single European Sky ATM Research Programme
SIP	Session Initiation Protocol
SJU	SESAR Joint Undertaking
SWIM	System-Wide Information Management
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Tag-Length-Value
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VDL	VHF DataLink

1 INTRODUCTION

1.1 Purpose

This document presents the design for a Controller–Pilot DataLink Communications (CPDLC) application within the ATMACA (Air Traffic Management and Communication over ATN/IPS) framework. It outlines the scope of the application, its intended operational context, supported CPDLC services, and the key system interactions aligned with current International Civil Aviation Organization (ICAO) and Single European Sky ATM Research (SESAR) communication strategies.

This document serves as a foundation for the development of the application, clarifying its components, interactions and requirements. It also guides future testing and integration within the ATMACA framework.

1.2 Scope

The main scope of this document is to detail the design of ATMACA's CPDLC application and to provide guidance for its future testing phase. It outlines the essential components of the application, including all the necessary information for the definition, interpretation, and processing of CPDLC messages, as well as a user guide to support its integration and testing within WP6.

1.3 Background

The Aeronautical Telecommunication Network (ATN) faces unique challenges due to its dynamic, high-mobility environment. Aircraft frequently transition between diverse coverage areas, networks, and communication technologies, including satellite and radio links. These transitions must meet stringent safety, reliability, and performance requirements, making robust communication protocols essential for supporting operational services.

ATMACA project proposes an Air Traffic Management (ATM) solution for ATN networks based on the Internet Protocol Suite (IPS), integrating a specialised datalink communication protocol, advanced operational applications (including DataLlink Context Management (DLCM) and CPDLC), an enhanced Human-Machine Interface (HMI), along with Green Route Operations (GRO) in air traffic control and management.

1.3.1 Mobility Management within ATMACA

The ATN must guarantee seamless data transmission, session continuity, mobility and availability, regardless of the characteristics of the operation. Beyond maintaining connectivity across multiple links, mobility management in the ATN must address three critical aspects: user mobility, session mobility, service mobility and terminal mobility. These four aspects are critical for supporting SESAR operations.

User mobility ensures that aircraft retain a consistent identity and access personalized services across various devices. Session mobility allows ongoing active communication sessions to continue without disruption as networks or devices change. Service mobility ensures that users can access the same applications and services across different networks and devices, preserving service quality even as connectivity conditions fluctuate.

The most promising protocols to address mobility challenges are PMIPv6 (Proxy Mobile IPv6), Ground-based Locator/ Identification (ID) Separation Protocol (GB-LISP) and Session Initiation Protocol (SIP) [16]. All of them have been analysed in detail in a previous deliverable (D2.1) to identify the limitations and advantages of the state of the art regarding mobility management.

Current protocols, such as PMIPv6 and GB-LISP, while offering certain advantages, have limitations that hinder their effectiveness in the dynamic and demanding aviation environment. Nonetheless, among its benefits, GB-LISP separates locators and identifiers, making it well-suited for managing mobility across different network domains, such as when aircraft move between networks of different providers. Conversely, PMIPv6 is better suited for intra-domain mobility, enabling seamless handovers within a single network without changing the mobile node's IP address. While GB-LISP can manage some intra-domain mobility, it is optimised for inter-domain scenarios.

These challenges and limitations are considered in the ATMACA solution design and development, and in the identification of use cases and operational improvements that ATMACA will make possible.

One of the most relevant aspects of ATMACA is mobility management, particularly in a multilink environment inherent to the ATN Network IP-based communication networks, as a critical aspect for SESAR operation. A multilink environment allows aircraft to connect to multiple communication datalink channels, such as satellite, terrestrial radio, and cellular networks, either simultaneously or in succession. In this environment, one of the primary challenges in the ATN is related with mobility, that is, ensuring uninterrupted communication between airborne avionics and ground-based systems while managing data exchange across diverse communication pathways.

To address mobility challenges, the ATMACA project will introduce a datalink communication protocol specifically designed to address the unique demands of the ATN. Inspired by the Diameter Base Protocol (RFC 6733) and the Session Initiation Protocol (SIP) (RFC 3261) [16], the ATMACA protocol adopts and extends their features to suit the complexities of aeronautical scenarios.

1.3.2 The ATMACA Framework

The framework shown in Figure 1.1 enables secure, reliable, and efficient communication, while allowing for seamless application integration and interoperability. Moreover, the HMI part enhances user interaction, providing intuitive control and situational awareness for air traffic operators. By combining these elements, the ATMACA solution delivers a scalable and future-proof system tailored to the evolving needs of aeronautical communication networks. It provides a scalable and adaptable foundation for integrating existing aeronautical applications and SWIM-enabled applications into ATN/IPS, while also supporting the development of new datalink applications.

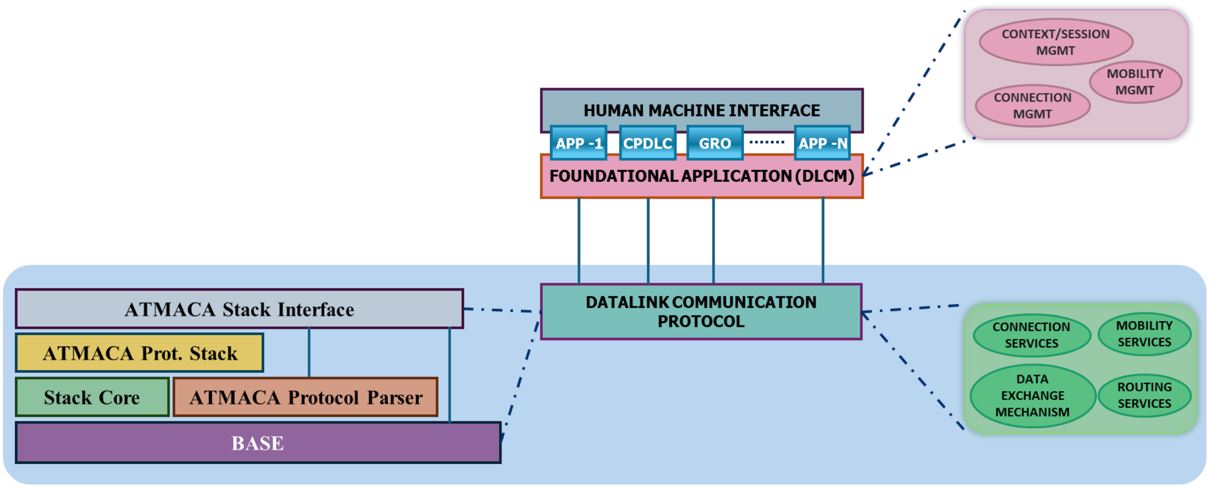


Figure 1.1: ATMACA Framework

With this development, the ATMACA project aims to achieve and showcase four key operational improvements in air traffic management (Figure 1.2). First, it seeks to streamline advanced Air Traffic Control (ATC) and communication handovers. Second, it focuses on providing fully flexible and customizable flight session management, enabling greater adaptability in managing flights. Third, it aims to ensure consistent and seamless datalink operations management, incorporating an enhanced HMI for both pilots and controllers. Finally, it will support trajectory prediction and trajectory improvement for Green Route Operations (GRO), enhancing the accuracy and efficiency of flight planning and execution. How CPDLC is covering operational improvements is discussed in detail in deliverable D2.2, ATMACA Functional Requirement Document (FDR).

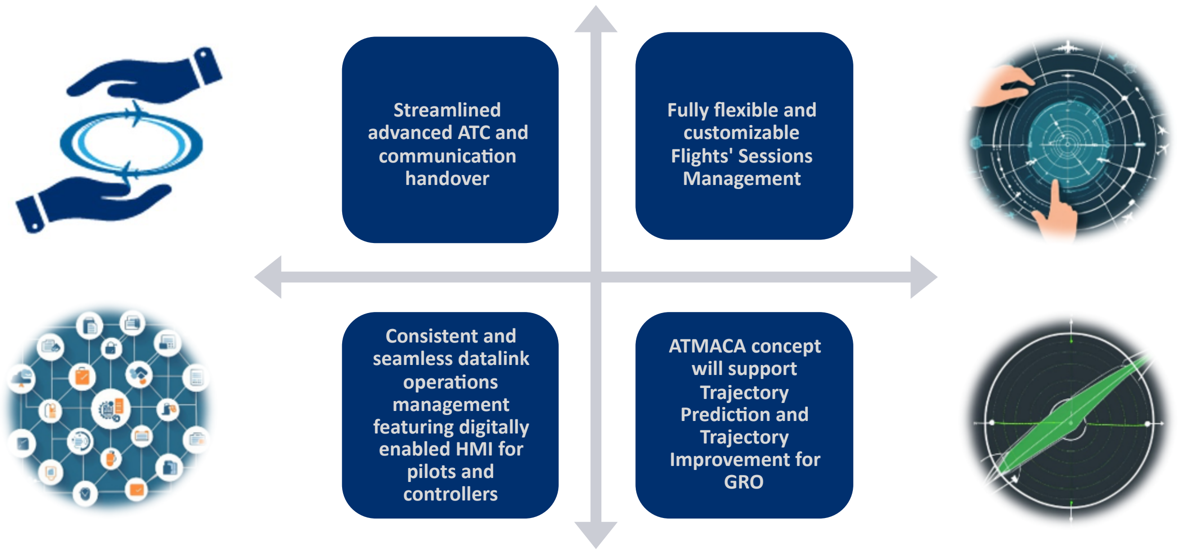

Streamlined advanced ATC and communication handover
Fully flexible and customizable Flights' Sessions Management
Consistent and seamless datalink operations management featuring digitally enabled HMI for pilots and controllers
ATMACA concept will support Trajectory Prediction and Trajectory Improvement for GRO

Figure 1.2: Key operational improvements aimed by ATMACA

1.3.2.1 DataLink Context Management Application

The ATMACA framework introduces a DLCM application (Figure 1.1) that enables seamless interaction between different aeronautical applications within the framework and the underlying protocol capabilities. DLCM acts as a middleware, managing context and session continuity, connection stability, and providing mobility support. Additionally, the ATMACA framework provides an interface for the development of new applications that can leverage protocol functionalities through context-based operations. By using this middleware, developers can create scalable and efficient applications that enhance air traffic management and communication services without modifying the core protocol.

1.3.2.2 Controller Pilot Datalink Application (CPDLC)

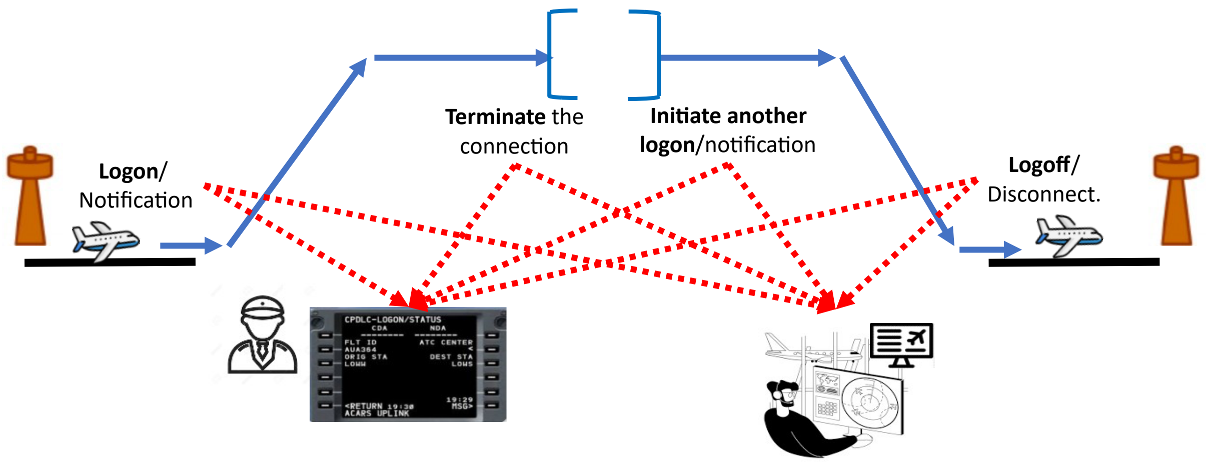
The CPDLC application is a software system designed to enable digital communication between air traffic controllers and flight crews (Figure 1.1). It supports the exchange of standardised messages over the ATN, in accordance with ICAO GOLD and relevant EUROCONTROL specifications.

This application provides an alternative and complementary channel to voice communications, aiming to reduce radio frequency congestion, enhance clarity and traceability of instructions, and support the evolution toward trajectory-based operations. It is particularly relevant in high-density airspace and for long-range en-route communications.

The CPDLC application is developed with a focus on interoperability, extensibility, and compliance with international standards. These goals are supported and enhanced with mobility, context and session management features provided by DLCM. Some of the key CPDLC features enabled by the framework and DLCM are detailed in the following sub-sections:

1.3.2.3 Seamless CPDLC Datalink operations

The use of datalink for Air Traffic Management (ATM) applications is growing rapidly, both in volume and variety. While several datalink services are already in operation and more are being developed, the manual handling of logon/logoff procedures (see Figure 1.3), as well as the management of disconnections and communication interruptions, can significantly increase the workload of pilots and Air Traffic Controllers (ATCos) if not properly managed.



```
CPDLC-LOGON/STATUS
      CDA          NDA
   --------     --------
FLT ID          ATC CENTER
AUA364                   <
ORIG STA          DEST STA
LOWW                  LOWS

                     19:29
<RETURN 19:30         MSG>
ACARS UPLINK
```

Figure 1.3: Manual Handling of datalink logon/off processes

With ATMACA, all datalink services and applications, including CPDLC are managed through a simple supportive interface. Pilots and ATCos will only need to open a session into the ATMACA system at the beginning of the flight or duty period. The system will automatically manage for them all logon/off, connections & disconnections across all required datalink services with a unified and fluid digital HMI.

The system automatically detects communication or service failures and resumes the connection as soon as possible without human intervention.

The session remains active throughout all flight phases, even during temporary signal losses (see Figure 1.4). This means that, although a connection may be interrupted, the session persists and automatically resumes once communication is restored, eliminating the need for repeated manual authentications by pilots and air traffic controllers. Additionally, this continuity is independent of the access technology and datalink service used, significantly benefiting long-haul flights (where communication technology changes frequently) and complex environments where switching between access technologies shall not disrupt service continuity. This functionality allows ATCos not only to access individual messages but also to retrieve the entire flight context at any time, regardless of connection status, enabling better decision-making based on historical and real-time information. This ensures that CPDLC communication remains available through all flight phases, creating a persistent, reliable and easy-to-use means of communication for pilots and controllers. By doing so, operations are simplified, workload is reduced along with use of voice communications.

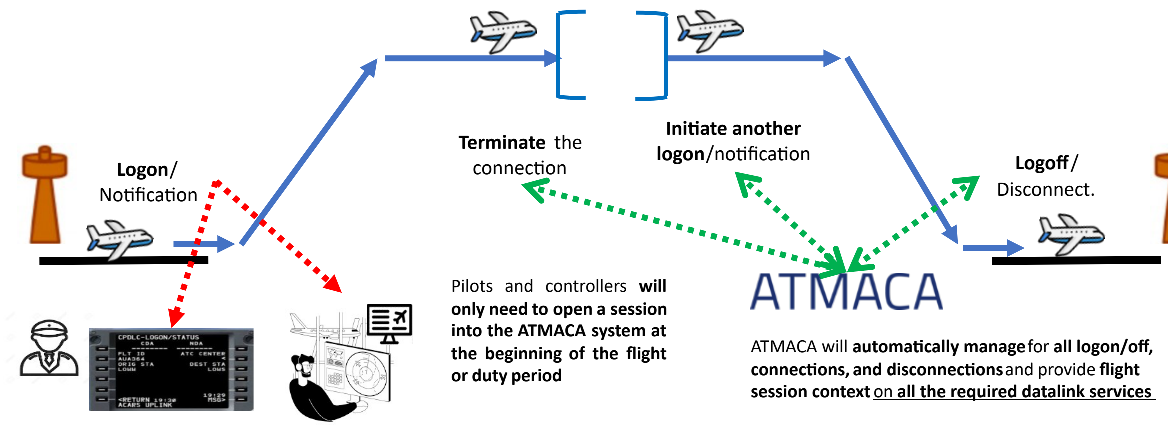

Logon/
Notification
Terminate the
connection
Initiate another
logon/notification
Logoff/
Disconnect.
Pilots and controllers will only need to open a session into the ATMACA system at the beginning of the flight or duty period
ATMACA
ATMACA will automatically manage for all logon/off, connections, and disconnections and provide flight session context on all the required datalink services

Figure 1.4: Automated Handling of datalink logon/off in ATMACA

1.3.2.4 Streamlined handover of flights

ATMACA allows for a flexible and seamless integrated management and handover between ATCos of all Air-Ground datalink communications, together with associated flight data and context information, over the ATN/IP communication infrastructure in a streamlined, automated, fast, cohesive, and synchronised manner. This Air Traffic Control (ATC) and communication handovers process will require minimal intervention from pilots and ATCos and will maximise flexibility in flight management and the allocation among ATCos. CPDLC will be used to coordinate and automate the process by ACM and DLIC services, reducing the need for manual intervention while maintaining situational awareness for both ATCos and pilots.

Furthermore, ATMACA enables flight transfer and handover from one controller to another with complete flexibility (Figure 1.5). ATMACA allows an ATCo to transfer all, some, or specific flights under his responsibility to any other ATCo, regardless of the ATCo's geographical location and area of responsibility, and regardless of whether ATC is based on a sector or flight flight-centric approach. This is possible thanks to the progression, updating, and management of the flight sessions that will evolve through the ATM system and the communication infrastructures as an IP communication-based flight session, while the flight progresses through the airspace. Moreover, automatic handover of flight sessions and the associated datalink hand off between ATCos will be supported by datalink communication procedure and specifically designed CPDLC messages.

Unique & univocal association between a flight and *all its Air-Ground datalink communications and historic,* any relevant flight data (planning, radar, trajectory, etc,..) *and context information, and its integration into **a single entity "flight session"***

*Progression, updating, and **management of the flight sessions*** that will evolve *through the ATM system and the communication infrastructures as an* **IP communication-based flight session,** while the flight progresses through the airspace

Automatic handover of **flight sessions** and the associated ***datalink hand off between*** ATCos supported by datalink communication procedure and specifically designed CPDLC messages.

Figure 1.5: Handover and Transfer of flights in ATMACA

1.3.2.5 Enhanced Context-Awareness through the Flight Session

ATMACA facilitates an Air Traffic Services (ATS) session-based system which offers greater operational flexibility for ATCos. ATMACA allows ATCos to group, aggregate, transfer, replicate, or divide flight sessions or responsibilities and implement new and flexible operational methods, in particular:

- **Display of presence information for ATCos and pilots:** ATMACA protocol provides presence information for aircraft, including being connected/disconnected and online/offline, and controllers, including their associated roles, workstations, and availability (Figure 1.6). After registration and logon, the system provides up-to-date status information for aircraft under control (and adjacent controllers) while maintaining situational awareness, including data on authorities' availability. This presence information is used to more easily access information required for CPDLC communications, as well as to present the status of the service. In addition, all information from previous CPDLC communications can be viewed and accessed in addition to being stored for future analysis or auditing purposes.

- **Capability to manage multiple sessions on a single workstation (managing Sessions for a role in a single ATC workstation).** The ATMACA solution allows ATCos, responsible for different operational domains (delivery, ground, tower), to manage concurrently multiple active sessions from a single workstation. Each session represents a specific aircraft and is displayed independently, providing a clear, organised view. This enables ATCos to handle a high volume of flights without losing operational clarity or compromising safety. The CPDLC application is role-based and provides different capabilities and services to each user according to their needs, in compliance with all relevant standards.

- **Managing sessions for a role in multiple ATC workstations.** It refers to ATCo's ability to replicate active sessions on multiple devices under their user account, such as a primary workstation, secondary workstation, or portable device (e.g., tablet or laptop), maintaining the same user. This ensures that the ATCo has uninterrupted access to all active session data, regardless of their physical location, allowing quick transition and immediate response in varying operational contexts.

- **Managing sessions for multiple roles in a single ATC workstation. (multiuser session aggregation).** The ATCo's workstation can host sessions from multiple domains, allowing the

ATCo to operate in different roles (ATC functions), such for example delivery, ground, and tower controller, within the same setup. Each session corresponds to an aircraft operating in one of these domains, giving the ATCo centralised control and visibility over activities typically managed by separate roles. (cross-domain Session Management).

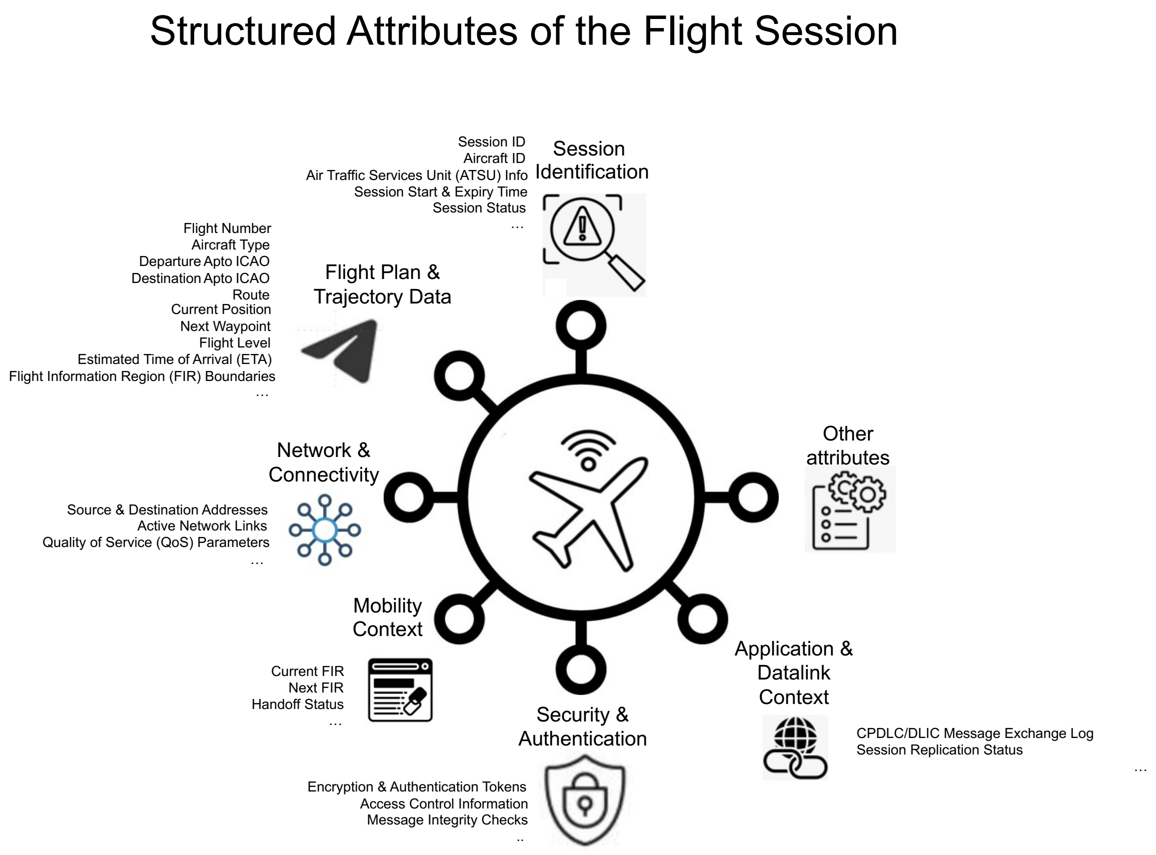

Structured Attributes of the Flight Session
Session Identification
Session ID
Aircraft ID
Air Traffic Services Unit (ATSU) Info
Session Start & Expiry Time
Session Status
...
Flight Plan & Trajectory Data
Flight Number
Aircraft Type
Departure Apto ICAO
Destination Apto ICAO
Route
Current Position
Next Waypoint
Flight Level
Estimated Time of Arrival (ETA)
Flight Information Region (FIR) Boundaries
...
Other attributes
Network & Connectivity
Source & Destination Addresses
Active Network Links
Quality of Service (QoS) Parameters
...
Mobility Context
Current FIR
Next FIR
Handoff Status
...
Application & Datalink Context
CPDLC/DLIC Message Exchange Log
Session Replication Status
...
Security & Authentication
Encryption & Authentication Tokens
Access Control Information
Message Integrity Checks
..

Figure 1.6: Structured Attributes of the Flight Session

1.4 Structure of the Document

This document has the following structure:

Section 1 (INTRODUCTION**):** outlines the purpose and scope of this document, along with; a brief background to support a better understanding of the following sections.

Section 2 (OVERVIEW OF THE ATMACA FRAMEWORK**):** provides an overview of the ATMACA framework and its base protocol.

Section 3 (CPDLC APPLICATION ARCHITECTURE**):** describes the CPDLC application architecture, its components, functions and interfaces with other ATMACA framework components.

Section 4 (CPDLC MESSAGE DEFINITION): lists the considered and implemented CPDLC messages, explaining their format and how they a built within the application.

Section 5 (APPLICATION REQUIREMENTS**):** provides all applicable requirements that define development.

Section 6 (USER GUIDE FOR INTEGRATION AND TESTING**):** provides a guideline to support the testing and integration phases within WP6.

Section 7 (REFERENCES**)**

2 OVERVIEW OF THE ATMACA FRAMEWORK

The ATMACA solution uses a modular architecture, with each layer addressing a specific functionality. The ATMACA datalink communication protocol uses a compact binary message format that operates independently of the underlying transport protocol.

A DLCM application is used to enhance session continuity, connection stability, and support mobility. The protocol's ability to efficiently manage communication and connections enables DLCM to act as a middleware, bridging network protocols with higher-layer applications.

2.1 ATMACA Network Architecture

The ATMACA network architecture, shown in

Figure 2.1, uses a layered structure where software-defined nodes are key elements and are responsible for managing communication and operational tasks. These nodes include:

- **ATM Server**: Manages critical network operations, user provisioning, registration, authentication, and authorisation.
- **Application Server**: Delivers specialised aeronautical services to stakeholders such as pilots, controllers, and dispatch teams.
- **ATC Agents**: Act as intermediary nodes, routing requests and maintaining seamless client-to-client and client-to-application server communications.
- **Context Management (CM) Agents**: manage role transitions, context registration, metadata synchronisation, and status tracking to ensure service continuity, scalability, and system resilience.
- **Clients**: Represent endpoints such as aircraft systems (mobile clients) and ATC workstations (stationary clients) that use ATMACA services.

In the case of CPDLC, the application does not run on an application server, but on client nodes. While applications such as CPDLC are primarily executed on client devices, services are hosted and delivered by Application Servers.

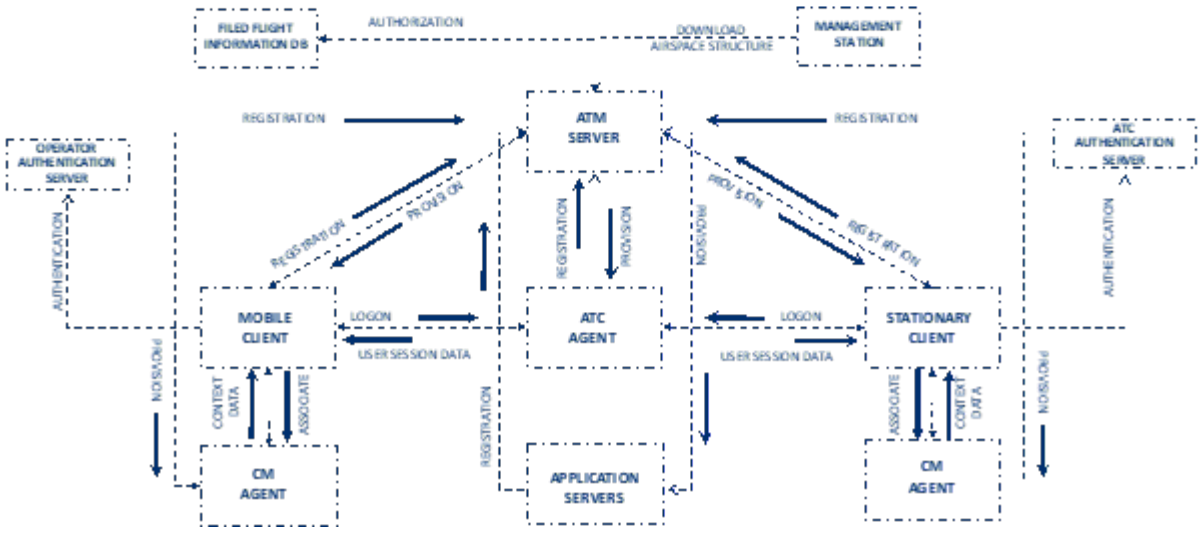


Figure 2.1: ATMACA's Network Architecture

2.2 Protocol Description

2.2.1 Software Infrastructure

- The ATMACA software is implemented in a layered structure in C++. The software infrastructure shown in
- Figure 2.2 is composed of:

  Base stack: implements building blocks such as Thread, Mutex, Socket, etc., where all OS specific differences are kept/handled in this layer. The base layer also provides logging, timer services, and other utilities that can be called or initialised by the user.

  ATMACA protocol parser stack: handles message creation and parsing. It provides methods and utilities for ATMACA message processing. With a dictionary approach, it creates a flexible and extensible mechanism for new ATMACA message handlers and Datalink Information eXchange (DIX) entries. There are two different dictionaries: the message and DIX dictionaries.

  The dictionaries used within the system are composed of maker classes, which are responsible for constructing both message objects and DIX handler objects. This design enables modularity and extensibility in the architecture.

  A key feature of this mechanism is its pluggable nature, which allows applications to register their own message and DIX handlers dynamically at runtime. This flexibility is essential for supporting evolving communication requirements in ATM environments.

  The DIX itself represents the data-carrying component of the ATM message protocol. It encapsulates the core information exchanged between system components, forming the foundation upon which higher-level message constructs are built.

  To support broad compatibility, the protocol parser is designed to be operating system independent. It leverages definitions provided by the protocol's base layer, enabling any layer of the system (including the application layer) to directly use the parser for constructing or interpreting messages.

- **Stack core**: implements a generic approach for network connections, connection management and message IO. This allows to implement of different protocols on top of it. It is not directly accessible by the applications.
- **ATMACA protocol stack**: implements ATMACA Protocol behaviour/requirements on top of Stack Core.

ATMACA stack interface: creates and manages all components, and it is based on the provided configuration. It can access protocol stack functionalities. The Stack Interface sets up threads, sockets, listening ports, and other resources based on configuration. All these are created in the stack core, and the ATMACA stack interface triggers them. On top of the stack interface, there are multiple applications (such as DLCM or GRO), which can access the protocol parser directly and also, by going through the stack interface, they can access the stack core functionalities. Additionally, the DLCM application provides extensive mobility, context, connection and session management features that can be used by other applications, acting as a middleware between these and the base protocol stack. This facilitates the integration with the protocol and simplifies the interactions and development process.

- **ATMACA application**: implements required functionality for a specific application. It directly uses the ATMACA Protocol Parser for message processing.

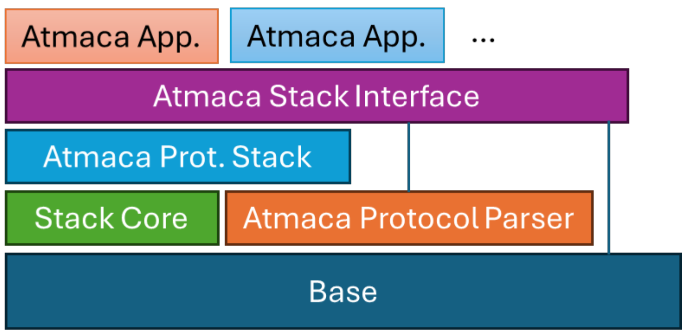


Figure 2.2. ATMACA's software architecture

2.2.2 ATMACA Protocol Message Format

- The ATMACA protocol message format, as seen in Figure 2.3, is inspired by the Diameter Base Protocol (RFC 6733), which uses Protocol Data Units (PDUs). ATMACA messages present the following format: Binary format messages, which consist of an ATMACA header and concatenated DIXs.

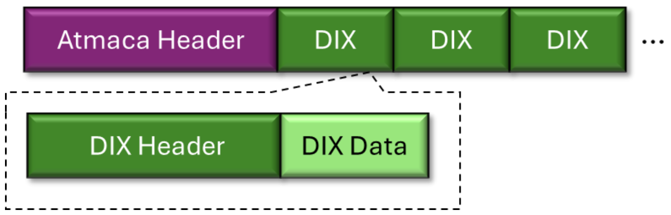


Figure 2.3: ATMACA message structure

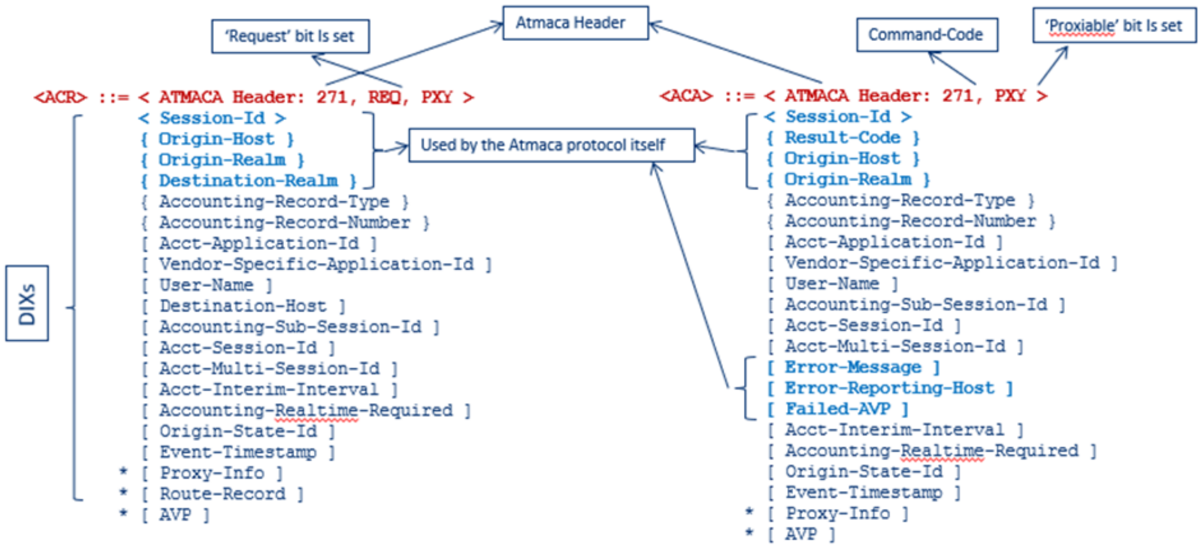
- The ATMACA header includes: Flags (version, priority, retransmission, request/response indicator), Length, Message code, Application ID (ATMACA header) and Request ID.
- DIXs are included in the message body, each of them having:
  - A header with flags (V, M, P), length, and optionally a Vendor ID
  - A data section with different possible data types (octet string, integer, float, unsigned, grouped (multiple DIX, sequence of DIXes), enumerate, address, time, etc).

The Vendor ID helps identify DIXs added by different organisations. For instance, UPM might include a vendor-specific DIX, and other vendors can do the same. This is useful when multiple vendors contribute to the same framework.

2.2.3 ATMACA Message Grammar Augmented Backus–Naur Form (ABNF)

To define and interpret messages consistently each DIX type is defined using the structure shown in Figure 2.4, defined by brackets:

- Angle brackets (<…>) indicate fixed position DIX elements.
- Curly brackets ({…}) mean the DIX is mandatory.
- Square brackets ([…]) mean the DIX is optional.
- (*) after an element means it can appear multiple times (e.g., 1* means at least one occurrence is required).



```
<ACR> ::= < ATMACA Header: 271, REQ, PXY >
          < Session-Id >
          { Origin-Host }
          { Origin-Realm }
          { Destination-Realm }
          { Accounting-Record-Type }
          { Accounting-Record-Number }
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ User-Name ]
          [ Destination-Host ]
          [ Accounting-Sub-Session-Id ]
          [ Acct-Session-Id ]
          [ Acct-Multi-Session-Id ]
          [ Acct-Interim-Interval ]
          [ Accounting-Realtime-Required ]
          [ Origin-State-Id ]
          [ Event-Timestamp ]
        * [ Proxy-Info ]
        * [ Route-Record ]
        * [ AVP ]
```

```
<ACA> ::= < ATMACA Header: 271, PXY >
          < Session-Id >
          { Result-Code }
          { Origin-Host }
          { Origin-Realm }
          { Accounting-Record-Type }
          { Accounting-Record-Number }
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ User-Name ]
          [ Accounting-Sub-Session-Id ]
          [ Acct-Session-Id ]
          [ Acct-Multi-Session-Id ]
          [ Error-Message ]
          [ Error-Reporting-Host ]
          [ Failed-AVP ]
          [ Acct-Interim-Interval ]
          [ Accounting-Realtime-Required ]
          [ Origin-State-Id ]
          [ Event-Timestamp ]
        * [ Proxy-Info ]
        * [ AVP ]
```

Figure 2.4: ATMACA ABNF

If a mandatory DIX is not in the dictionary, it can be implemented. Otherwise, if the mandatory bit is not set and the DIX is unknown, it will be ignored.

2.2.4 ATMACA Messaging Concept

ATMACA is a peer-to-peer protocol where any node can initiate a request. Each request/response pair is encoded as a transaction in transaction-based messages or within a session in session-based messages. Communication between ATMACA peers begins with one peer sending a message to another ATMACA peer, which is called a "request". This starts a "transaction" or is made through the session, waiting for a "response". The transaction is ended/completed when the sender receives the corresponding response.

A "session", which is a related progression of events devoted to a particular activity, may consist of multiple request/response pairs. A session is identified by the same session ID included in all messages. The ATMACA messaging session is activated between two nodes upon the first request and identifies all communications (request and response messages) interchanged between these nodes.

When an application starts a session, it does it with a dedicated message. In the case of CPDLC, for example, it is going to be a CPDLC connect message. From that point, a session ID is created and maintained for the duration of that specific interaction between those two ends. This session ID helps to identify and track the ongoing communication within that session. On the other hand, the registration and log-on is performed by DLCM.

There are two types of messaging:

- Transaction-based message: request and response transaction. Each interaction is self-contained and then closed. These might not need a persistent session ID.
- Session-based message: once the session is established, the communication is application dependent, which is the case of CPDLC.

GRO might be used with transaction-based or session-based messaging, based on the application designer's decision, while CPDLC should use session–based messaging.

2.2.5 ATMACA Message Processing

There are 2 types of dictionaries:

- DIX dictionary, based on DIX-code and Vendor-ID
- Message dictionary, based on ATMACA app ID, Command code, and request indicator values.

Each dictionary has a maker. The maker provides the mechanism to create an appropriate handler. The handler may be specific to a data component or may implement a common behaviour for a group of data components.

To build an ATMACA Protocol Data Unit (PDU), the application needs to provide "message data". The result gives raw data formatted as ATMACA PDU.

To parse an ATMACA PDU, the application applies the PDU as raw data and gets "message data" as an output produced from the PDU.

3 CPDLC APPLICATION ARCHITECTURE

3.1 Architecture Overview

In ATMACA, CPDLC is treated as a specialised application that operates on top of the DLCM application, utilising context and session management to ensure the persistence, integrity, and traceability of all CPDLC interactions. The CPDLC application services, shown in Figure 3.1, support both ground and airborne deployment scenarios, including controller-pilot messaging, service mobility across ATC agents, and automated message routing between operational sectors.

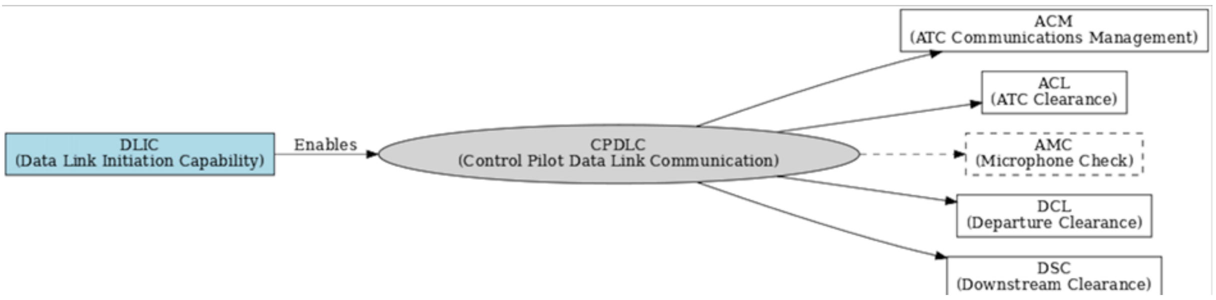


Figure 3.1.CPDLC Service architecture

At the foundation of CPDLC lies the DataLink Initiation Capability (DLIC, or DLCM in this case, which is responsible for initiating the operational datalink presence between aircraft and the ground system. Within the ATMACA architecture, the DLIC is part of the DLCM application

3.2 CPDLC's Role and Purpose

CPDLC is a core application layer in the ATMACA architecture, designed to support standardised air-ground communications between pilots and controllers using digital messaging. Its main role is to facilitate routine ATC communications that would otherwise be handled via voice, offering increased efficiency, reduced frequency congestion, and support for long-range communications, especially over oceanic and remote areas.

In ATMACA, the CPDLC module builds upon the foundational DLIC exchange, which establishes operational readiness and security parameters. -With DLIC, the module enables various higher-level CPDLC services such as ACM (ATC Communications Management), ACL (ATC Clearance), DCL (Departure Clearance), and DSC (Downstream Clearance). These services support key operational workflows such as frequency changes, flight level assignments, route amendments, departure coordination, and coordination across FIR boundaries.

The client node, hosting the CPDLC module, is designed to work in coordination with the ATC Agent, which manages message routing, session binding, and delivery guarantees between airborne clients and ground systems. Each CPDLC transaction is securely tied to an existing DLIC context, ensuring traceability, validation, and integrity for every exchanged message.

3.3 CPDLC's Main Functional Requirements

The CPDLC module in ATMACA must support a reliable and extensible exchange of controller-pilot messages aligned with ICAO GOLD specifications.

MUST Have:

- Support for sending and receiving ACL (ATC Clearance) and ACM (ATC Communication Management) message sets.
- Support for DCL (Departure Clearance).
- DLIC functionality must be fully established before initiating any CPDLC services, as handled by the DLCM Module (see Section 5.4.3).
- Session association and routing through the ATC Agent without interpreting CPDLC content (application-transparent relay).
- Validation of CPDLC messages based on context ownership, active session state, and destination availability.
- Logging of all CPDLC message activity (incoming and outgoing) for traceability, audit, and recovery.

SHOULD Have:

- Retry and timeout handling for CPDLC messages through the ATC Agent in case of delivery failure.
- Integration with Session Manager for dynamically binding CPDLC messages to existing sessions.
- Role-based validation of message types to ensure compliance with operational rules.

COULD Have:

- DSC (Downstream Clearance) message types.
- Message prioritisation or queuing support based on operational urgency.
- Feedback indicators for CPDLC message delivery state (e.g., pending, delivered, acknowledged).
- Support for session-based analytics (e.g., number of messages per session, delay metrics).

WON'T Have:

- Support for AMC (ATC Microphone Check) in the initial release.
- Direct peer-to-peer CPDLC messaging between clients (must always route through ATC Agent).
- Interpretation or modification of CPDLC payloads at the ATC Agent level.

3.4 CPDLC Architecture and Framework Interactions

The CPDLC module in the ATMACA system is architected as a modular, session-bound application layer that leverages the underlying DLCM services, namely context, session, and connection management. It enables structured, traceable, and secure controller-pilot data exchanges that comply with ICAO GOLD standards.

Every CPDLC transaction, whether a clearance, communication assignment, or departure message, is strictly bound to a validated DLIC context and an active session, ensuring consistent routing and context ownership.

Architecturally, it is designed to maintain a persistent session between air and ground participants throughout the lifespan of CPDLC communications. Every CPDLC transaction—whether it's ACL, ACM, DCL, or DSC—is strictly bound to an active session context established between an aircraft and a ground controller.

The design enforces that no CPDLC service is processed outside of an active session, ensuring operational continuity, secure delivery, and protocol adherence. This fully session-based architecture supports controller and pilot mobility, service migration, and long-duration exchanges that span airspace boundaries.

CPDLC messages are routed via the ATC Agent, which transparently forwards them between endpoints, handling delivery retries without interpreting content semantics. This session-bound design supports seamless controller/pilot mobility, message continuity across airspace boundaries, and persistent interaction throughout the duration of the flight.

To implement the CPDLC application, the following functional components and interactions within ATMACA, shown in Figure 3.3, have been defined.

ATMACA Stack Interface: it provides a foundation for application developers which handles the communication flow and interchange of messages. It allows access to the ATMACA protocol parser, which handles message creation and parsing, essential for CPDLC message interchange.

DLCM: The DLCM application plays a central role in how ATMACA manages communication across air and ground systems. It brings together four main modules: context, session, connection, and mobility Management, each handling a specific part of the system's operation. The context management module keeps track of user roles, system state, and operational assignments, helping coordinate services across different nodes. The session management module oversees starting, maintaining, and ending communication sessions, ensuring that data flows smoothly even during transitions. The connection management module handles the network links themselves, checking their health, recovering from failures, and keeping everything stable. Finally, the mobility management module ensures that communication continues without interruption when users or systems move across different regions or networks. Together, these modules form the backbone of DLCM, allowing ATMACA to deliver reliable, flexible, and context-aware communication in complex, changing environments.

The connection management module provides traditional DLIC service. It is executed prior to the first use of the CPDLC application. It consists of the Logon and Contact functions, establishing the connections needed for the CPDLC application.

HMI: the human-machine interface displays all CPDLC and ATMACA information to end users in a simple and effective way. The CPDLC application will provide all required CPDLC message data to it and interface with it to allow users to write, send and receive messages and access flight session information, through an easy-to-use graphic interface.

The CPDLC application itself has the following components, as shown in Figure 3.2:

- CPDLCServiceHandler – The central control unit for CPDLC operations. It handles both sending and receiving workflows, validates message structure, and routes messages to the appropriate subsystem.
- SessionManagementInterface – An integration point to the DLCM Session Manager for ensuring messages are associated with an active session context. It provides session lookup, validation, and binding APIs.
- MessageBuilder – Responsible for constructing outgoing CPDLC messages based on operational data and application logic. It ensures compliance with GOLD-standard message formats.
- MessageProcessor – Parses, validates, and handles incoming CPDLC messages. It coordinates with the CPDLCServiceHandler and ensures message semantics are correctly interpreted.
- AuditLogger – Records CPDLC transactions for traceability, fault diagnosis, and compliance. Every sent and received message is logged with metadata such as session ID, timestamp, and delivery state.

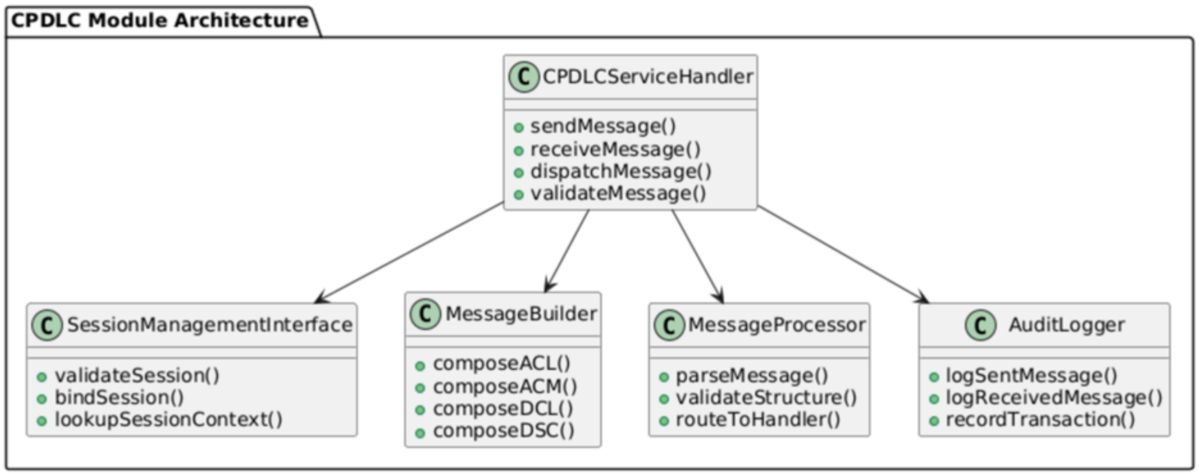


Figure 3.2: CPDLC's Module architecture

Following the previously described modules and functions the following diagram in Figure 3.3 represents the CPDLC's architecture and its connections to other ATMACA applications and the base protocol stack interface.

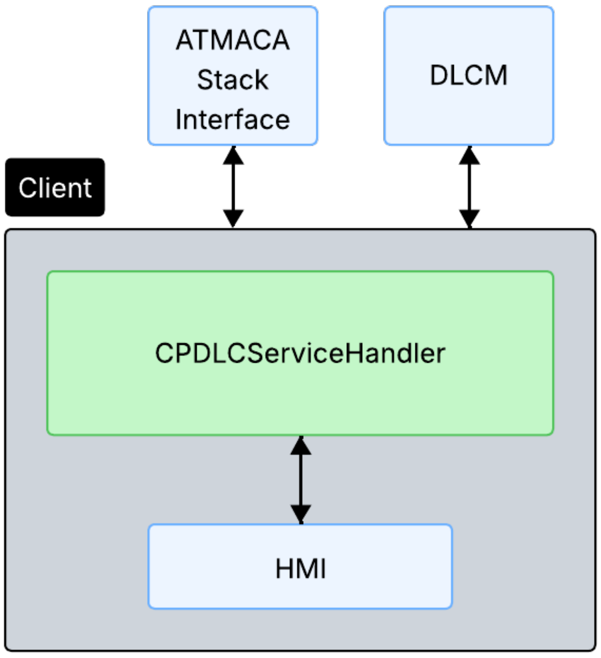


Figure 3.3: CPDLC's framework interactions

3.5 Functional Subsystem

The functional subsystem layer of the CPDLC module defines the dynamic coordination logic responsible for executing messaging workflows in real time. These subsystems interact with the underlying architectural components to process both outbound and inbound messages while maintaining protocol compliance, session awareness, and full operational traceability. The CPDLC functional logic is modularised into five key subsystems, as shown in Figure 3.2:

- **Outbound Processing** Subsystem – Responsible for composing, validating, and dispatching outbound messages.
- **Inbound Processing Subsystem** – Handles parsing, validation, and routing of incoming messages to appropriate application logic.
- **Session Validation & Binding** Subsystem – Ensures that every CPDLC message is linked to an active session and validates ownership.
- **Audit & Traceability Subsystem** – Records all message transactions for compliance, diagnostics, and fault investigation.

Together, these subsystems support the safe and consistent execution of CPDLC workflows across distributed air-ground deployments, while enabling message reliability, operational resilience, and regulatory compliance.

Table 3.1presents the mapping between the functional subsystems of the CPDLC module and their corresponding architectural elements, illustrating how each runtime component interacts with the static infrastructure to execute CPDLC messaging workflows. The table highlights how outbound and inbound message flows, session validation, and audit logging are distinctly modularized yet interdependent. These relationships ensure that all CPDLC communications are securely bound to active sessions, properly validated, and fully traceable across operational scenarios.

Table 3.1: CPDLC Module - Linkage Table

Functional Subsystem	Related Architectural Elements	Interaction Summary
Outbound Processing Flow	CPDLC Service Handler, Message Builder, Session Management Interface, Audit Logger	Composes and sends CPDLC messages, validates session context, and logs outbound operations.
Inbound Processing Flow	Message Processor, Session Management Interface, Audit Logger	Parses and validates incoming messages, verifies session association, and records reception logs.
Session Validation & Binding	Session Management Interface	Resolves session ownership, performs lookup and binding for CPDLC message association.
Audit & Traceability Subsystem	Audit Logger	Records all sent/received message events, including metadata for traceability and diagnostics.

Figure 3.4 depicts the class-level architecture of the CPDLC module, showing the interaction between functional subsystems (e.g., outbound processing, session validation) and key architectural components (e.g., message builder, service handler, audit logger). This Unified Modelling Language (UML)-based view provides a structured representation of how modular responsibilities are coordinated to support compliant, session-aware controller–pilot datalink communications within the ATMACA framework.

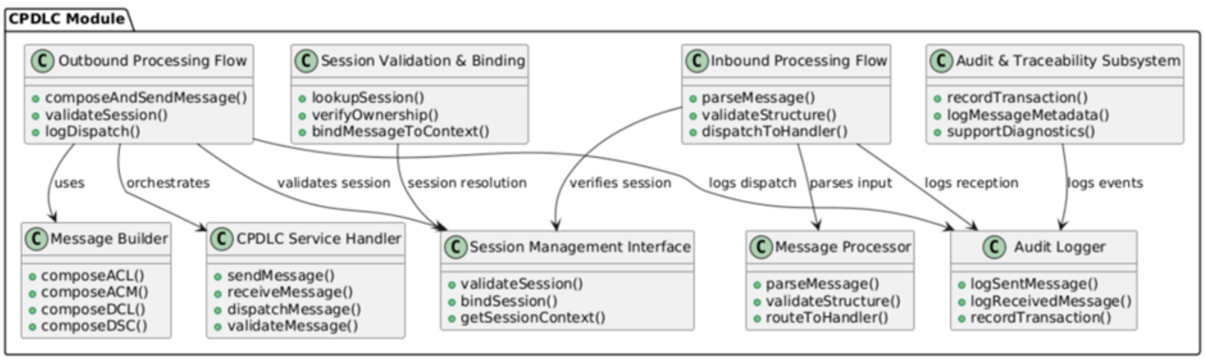


Figure 3.4: CPDLC Core Module - Class - Functional Architecture Level

The CPDLC module in ATMACA supports a suite of standardised message services defined by ICAO GOLD and operational air traffic control procedures. These services are implemented as modular extensions of the CPDLC core architecture, leveraging the same session-aware message handling pipeline while providing specialised logic for operational scenarios such as departure coordination, altitude clearance, frequency changes, and trajectory pre-coordination.

For documentation purposes, Datalink Services are grouped into two categories (Groups 1 and 2):

3.5.1 Group 1 – Standard Datalink Services

These include well-established message types such as Departure Clearance (DCL), ATC Communication Management (ACM), ATC Clearance (ACL), and Downstream Clearance (DSC). These services are fully defined and implemented within the current ATMACA release.

3.5.1.1 Departure Clearance (DCL)

The Departure Clearance (DCL) service (Figure 3.5) supports digital delivery and acknowledgement of pre-departure clearances, such as route, squawk code, and initial altitude assignments. Within ATMACA, DCL operates over the CPDLC core architecture, enabling both ATC-originated requests and aircraft-generated responses. The service is capable of sending DCL messages, receiving and processing inbound requests (e.g., from test tools or training environments), composing standardised responses (ACK, REJ), and handling acknowledgements returned by the aircraft. All message transactions are session-bound and logged, ensuring traceability and consistency across pre-departure workflows.

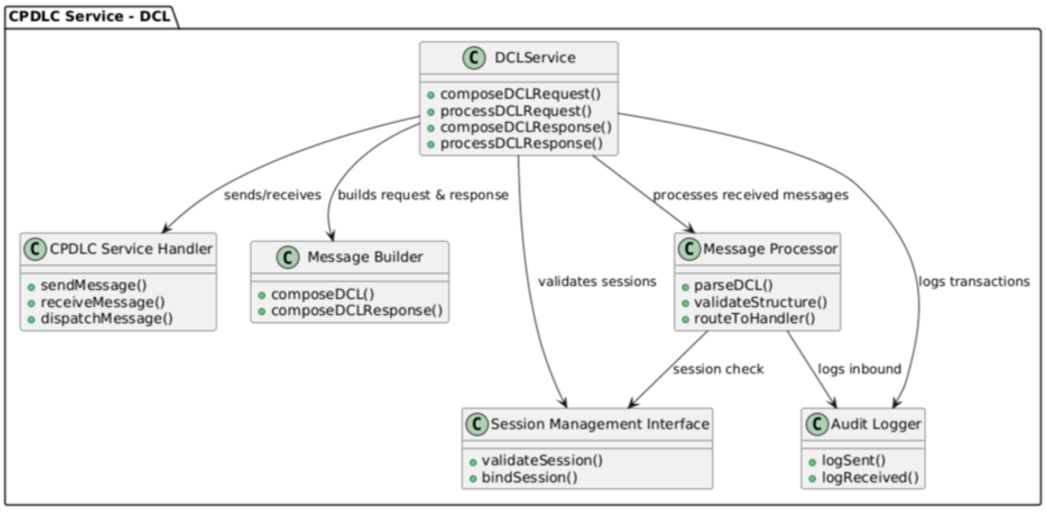


Figure 3.5. DCL Service - Class Level Architecture

3.5.1.2 ATC Communication Management (ACM)

The ACM service (Figure 3.6) enables the exchange of communication transfer instructions between ATC and pilots during controller handovers. This includes frequency changes and contact instructions. In ATMACA, ACM supports a full lifecycle wherein the originating system composes and dispatches an ACM request, the receiving end validates and processes it, and if applicable, generates a structured response. This two-way interaction is orchestrated through the CPDLC Service Handler and relies on the Message Processor and Builder for dynamic message flow handling, session validation, and event logging.

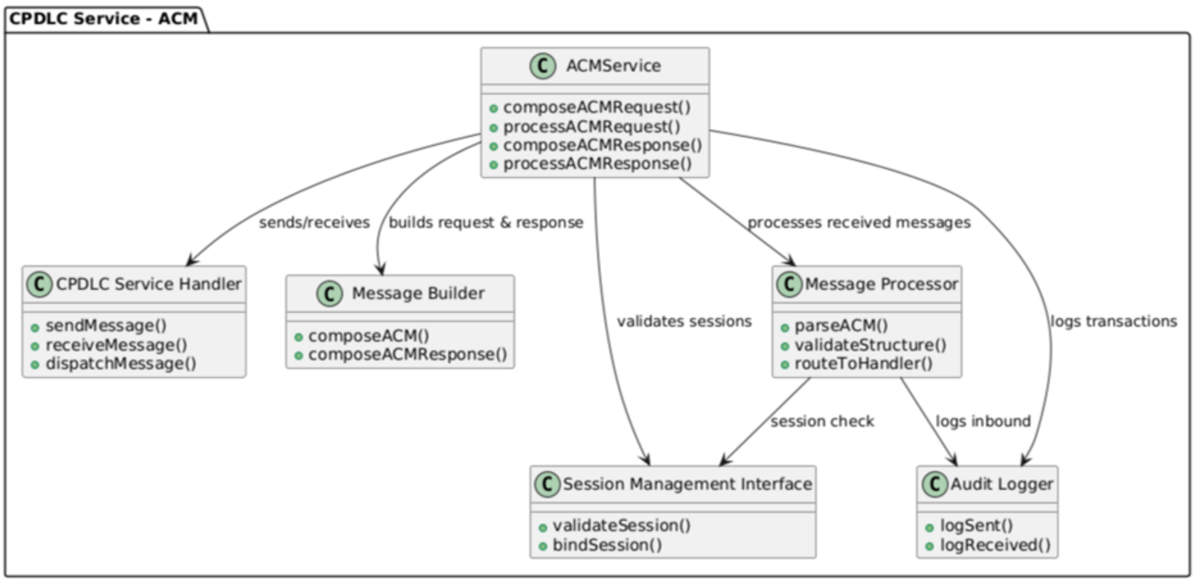


Figure 3.6. ACM Service - Class - Level Architecture

3.5.1.3 ATC Clearance (ACL)

The ACL service (Figure 3.7) facilitates in-flight air traffic control clearances, including route changes, flight level assignments, and procedural directives. The service supports bidirectional exchange, allowing both ATC-originated clearance messages and aircraft-generated requests or replies. The ACL message cycle includes message composition, session-bound delivery, receipt processing, response generation, and logging. ATMACA's modular CPDLC architecture ensures that all ACL interactions are validated, acknowledged, and archived for operational integrity and compliance with ICAO GOLD standards.

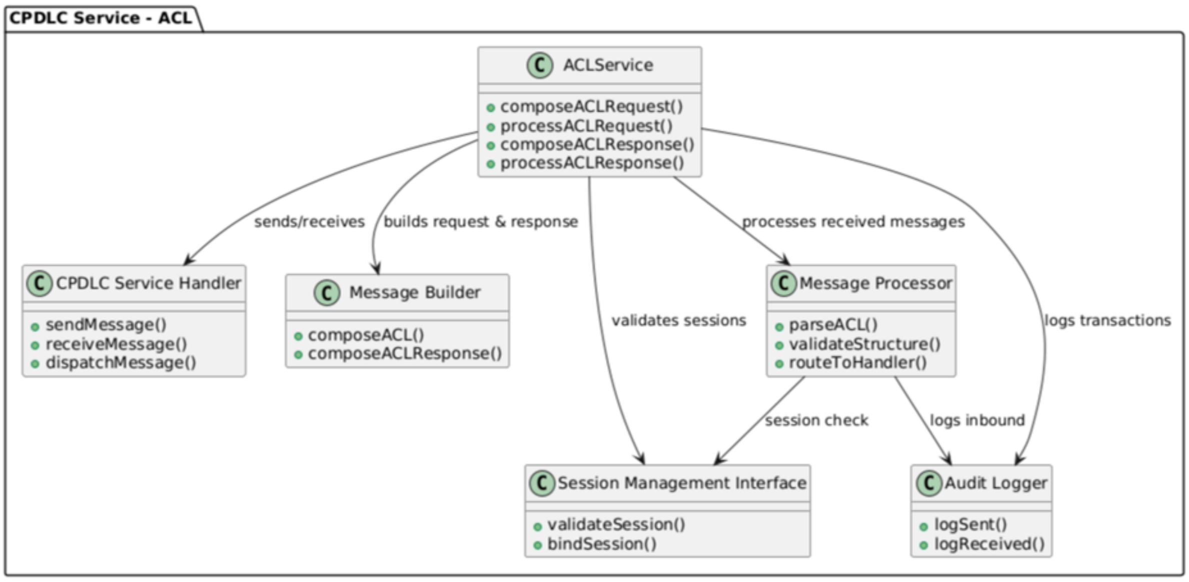


Figure 3.7.ACL Service - Class-Level Architecture

3.5.1.4 Downstream Clearance (DSC)

The DSC service (Figure 3.8) provides a mechanism for issuing pre-authorised clearances from downstream control units or future airspace sectors. Within ATMACA, DSC functions as a full-duplex

EUROPEAN PARTNERSHIP

Co-funded by the European Union

service, enabling ATC to schedule and send downstream instructions while enabling aircraft systems to receive, validate, and respond to these directives. It supports proactive clearance planning and cross-sector coordination, utilising the same CPDLC message flow framework for building requests, receiving replies, and managing acknowledgement cycles. Additional support for delivery scheduling is provided through the DSCService logic layer.

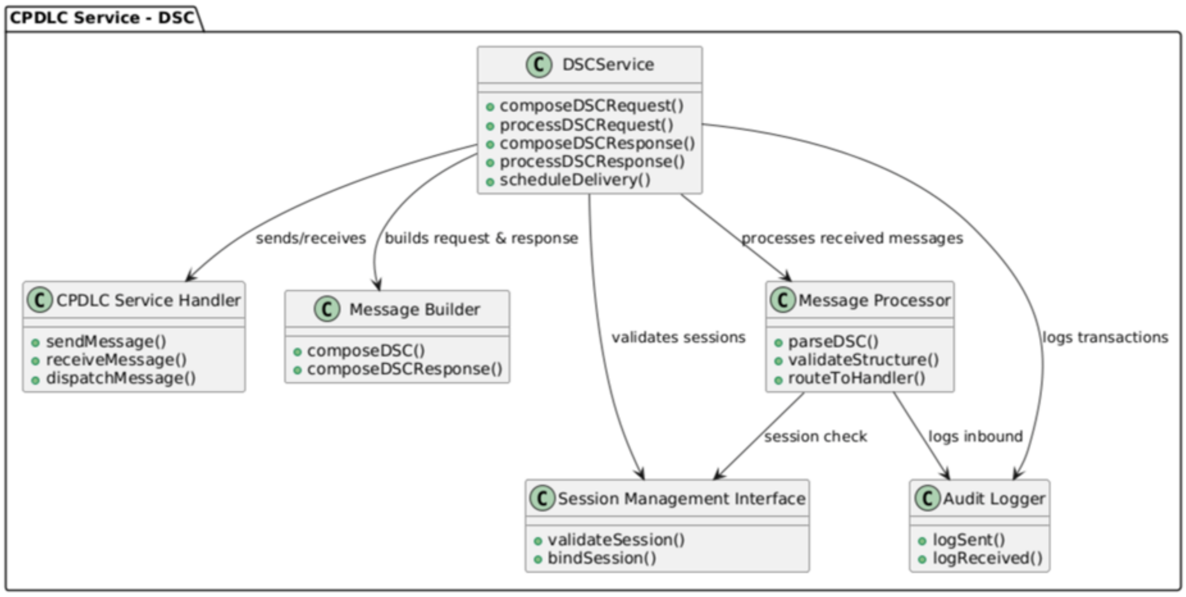


Figure 3.8. DSC Service - Class-Level Architecture

3.5.2 Group 2 – Enhanced Ground Services

The CPDLC module in ATMACA is designed with extensibility in mind, allowing new service types to be integrated seamlessly into its session-based messaging architecture. This section outlines a second group of CPDLC services—D-ATIS, D-TAXI, D-PUSHBACK, and D-STARTUP—which focus on digitalising ground operations through controller–pilot datalink.

These services are not yet implemented in the current release but are defined within the system's architectural roadmap. They will leverage the same core subsystems—such as the CPDLC Service Handler, Message Builder, Session Management Interface, and Audit Logger—to provide traceable, session-aware communication during ground phases of flight operations.

Each service will follow a full request-response lifecycle and support automation triggers, allowing controllers to issue ground instructions and receive structured acknowledgements from aircraft systems. The subsections below describe the planned behaviour and system integration for each enhanced service.

3.5.2.1 D-ATIS (Digital Automatic Terminal Information Service)

D-ATIS (Figure 3.9) provides real-time digital broadcasts of terminal information—such as weather, active runways, and NOTAMs—directly to aircraft via CPDLC. Instead of receiving this information via VHF broadcast, pilots will retrieve the latest ATIS through structured messages over an established datalink session.

In the planned ATMACA implementation, D-ATIS messages will be generated automatically by the ground system or requested by the flight crew. The service will rely on the MessageBuilder to compose standardised reports, which will be session-bound and dispatched via the CPDLC Service Handler. Pilots may acknowledge receipt or send ATIS version mismatch reports. All inbound and outbound messages will be logged by the Audit Logger and managed within the existing session lifecycle.

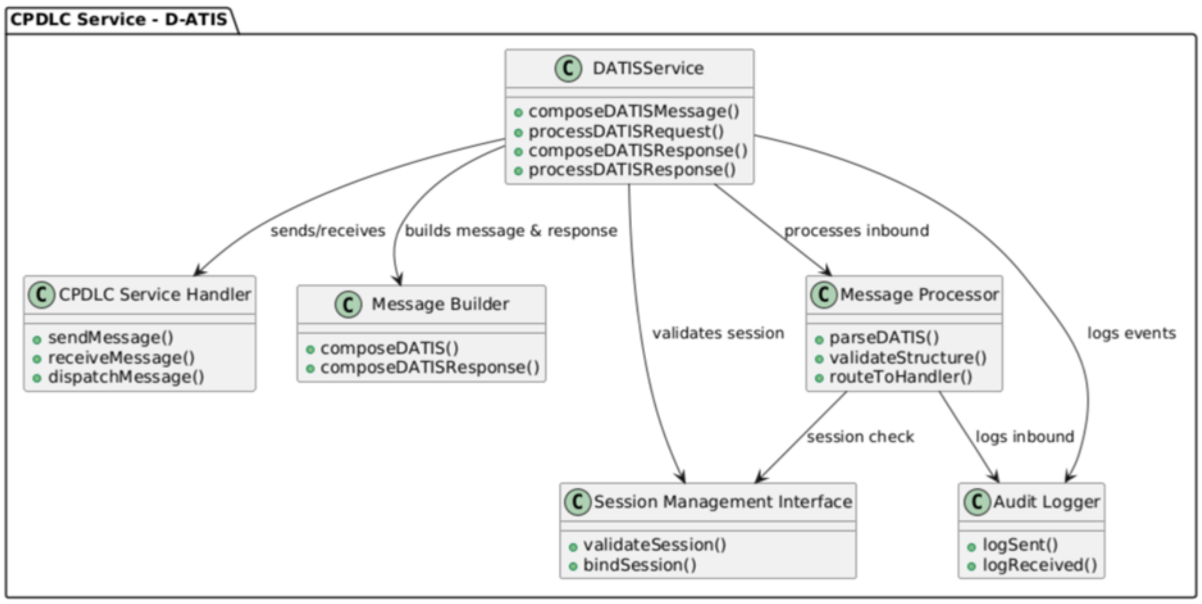


Figure 3.9: D-ATIS Service - Class-Level Architecture

3.5.2.2 D-TAXI (Digital Taxi Clearance)

D-TAXI (Figure 3.10) enables controllers to issue taxi route clearances digitally, reducing voice congestion and increasing operational clarity during high-density surface operations. It will allow dynamic route instruction, including hold points, intersections, or updated runway assignments.

D-TAXI in ATMACA will support both ATC-issued taxi instructions and pilot readback acknowledgements using structured messages. The composeDTAXI() and parseDTAXI() methods will be added to the MessageBuilder and MessageProcessor, respectively. The service will be coordinated through the CPDLC Service Handler and validated against session ownership to ensure the instructions are delivered to the correct aircraft.

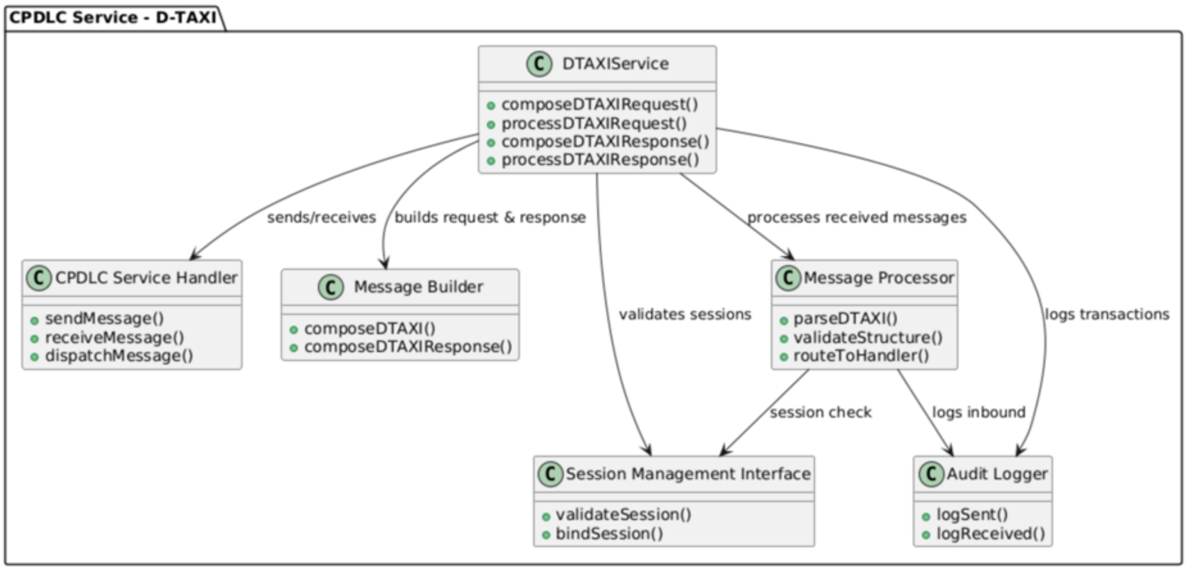


Figure 3.10: D-TAXI Service - Class-Level Architecture

3.5.2.3 D-PUSHBACK

D-PUSHBACK (Figure 3.11) allows pilots to request pushback approval and receive confirmation via CPDLC. It simplifies pre-taxi procedures and provides timestamped records for ramp movement coordination.

In ATMACA, the service will support both pilot-initiated pushback requests and controller responses through the full request-response lifecycle. composePushbackRequest() and composePushbackClearance() methods will be available in the MessageBuilder, while acknowledgments and errors will be handled by the MessageProcessor. The session link ensures that ground control instructions are issued within an authorised operational context.

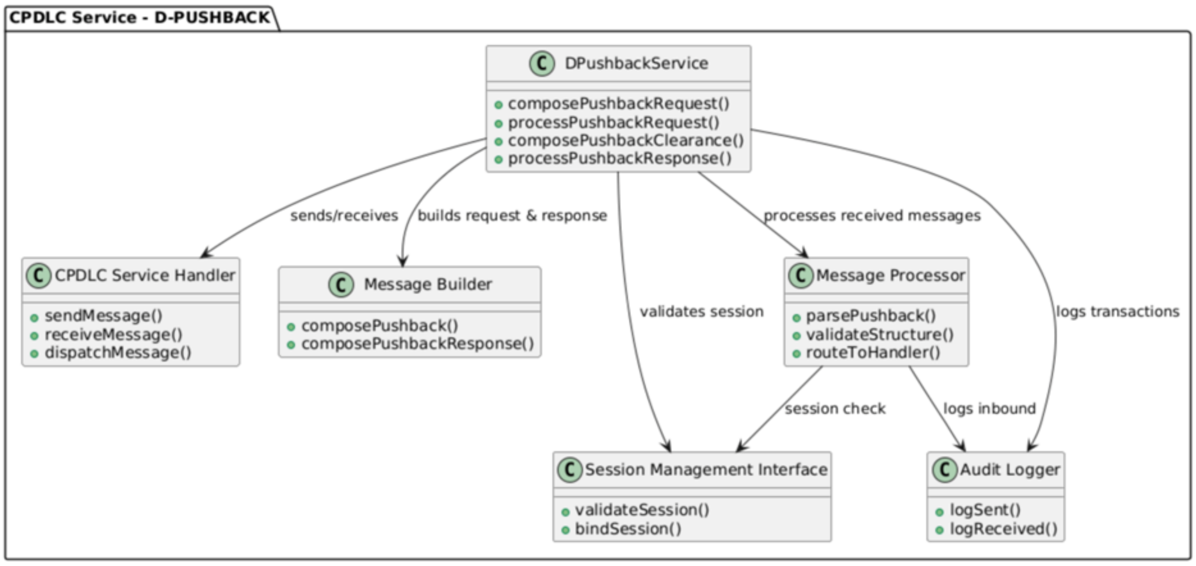


Figure 3.11: D-PUSHBACK Service - Class-Level Architecture

3.5.2.4 D-STARTUP

D-STARTUP (Figure 3.12) manages the clearance exchange for engine startup procedures prior to taxi and departure. It ensures synchronisation between ATC and aircraft, particularly in congested apron or ramp scenarios.

ATMACA's D-STARTUP service will enable the aircraft to digitally request startup and receive approved startup instructions or delay messages from the controller. This interaction will be managed by the CPDLC core pipeline, with extensions to the Builder and Processor components. It supports compliance tracking and improves predictability in gate release and pre-taxi sequencing.

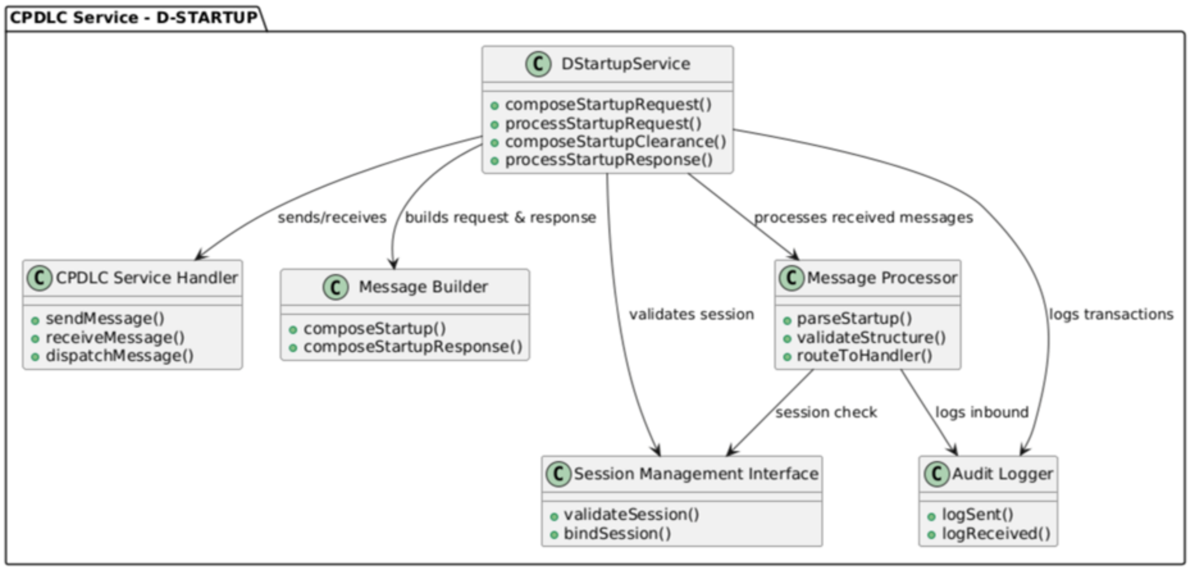


Figure 3.12: D-STARTUP Service - Class-Level Architecture

3.6 Datalink Services Workflow

The Datalink Services Workflow in ATMACA supports two operational modes based on the underlying transport protocol: Strict Pair Mode (UDP) and Hybrid Mode (TCP). In Strict Pair Mode, which is used over UDP-based communication, the system enforces a rigid request–response structure where every CPDLC message, including acknowledgements and operational responses, must be followed by a dedicated return message. Even response messages are transmitted as standalone CPDLC requests and require their own dedicated responses to complete the communication cycle. This ensures message integrity and traceability in environments where packet delivery is not guaranteed. In contrast, the Hybrid Mode (TCP) leverages the reliability of TCP to streamline the workflow. In this mode, response messages may be implicitly acknowledged or may carry additional operational content directly within the response payload, significantly reducing message overhead while maintaining consistency. This dual-mode approach provides flexibility for deploying ATMACA over a wide range of operational and network conditions.

3.6.1 Predeparture Clearance (DCL)

The Pre-Departure Clearance services in ATMACA encompass all controller–pilot datalink communications issued before the aircraft enters the active runway. This includes both standard ICAO-compliant Departure Clearance (DCL) messages and a set of extended, ATMACA-specific services such as D-STARTUP, D-PUSHBACK, and D-TAXI. All of these services are transmitted using the shared

EUROPEAN PARTNERSHIP

Co-funded by the European Union

DCL_COMMAND envelope and follow ATMACA's structured request–response model. This unified approach allows for seamless integration of custom pre-flight procedures alongside traditional DCL messaging while maintaining full compatibility with both UDP and TCP communication environments.

3.6.1.1 Departure Clearance Service

The Departure Clearance (DCL) service in ATMACA facilitates the digital exchange of pre-departure instructions between air traffic controllers and flight crews. In alignment with ICAO CPDLC standards, the system supports the transmission of clearances such as route assignments, altitude constraints, and squawk codes using DCL_COMMAND messages.

The DCL workflow represents a structured digital communication sequence between the aircraft (pilot) and the ATS Unit (controller). The interaction starts when the pilot issues a departure clearance request via the CPDLC system. The ATSU may respond with a STANDBY if additional processing time is needed. Once the clearance is available, the ATSU sends it digitally to the aircraft. The pilot can either accept (WILCO) or reject (UNABLE) the clearance. Logical acknowledgements (LACKs) may be exchanged as required by local procedures to confirm message receipt. This workflow minimises voice communication, reduces potential misunderstandings, and enhances automation in the pre-departure phase.

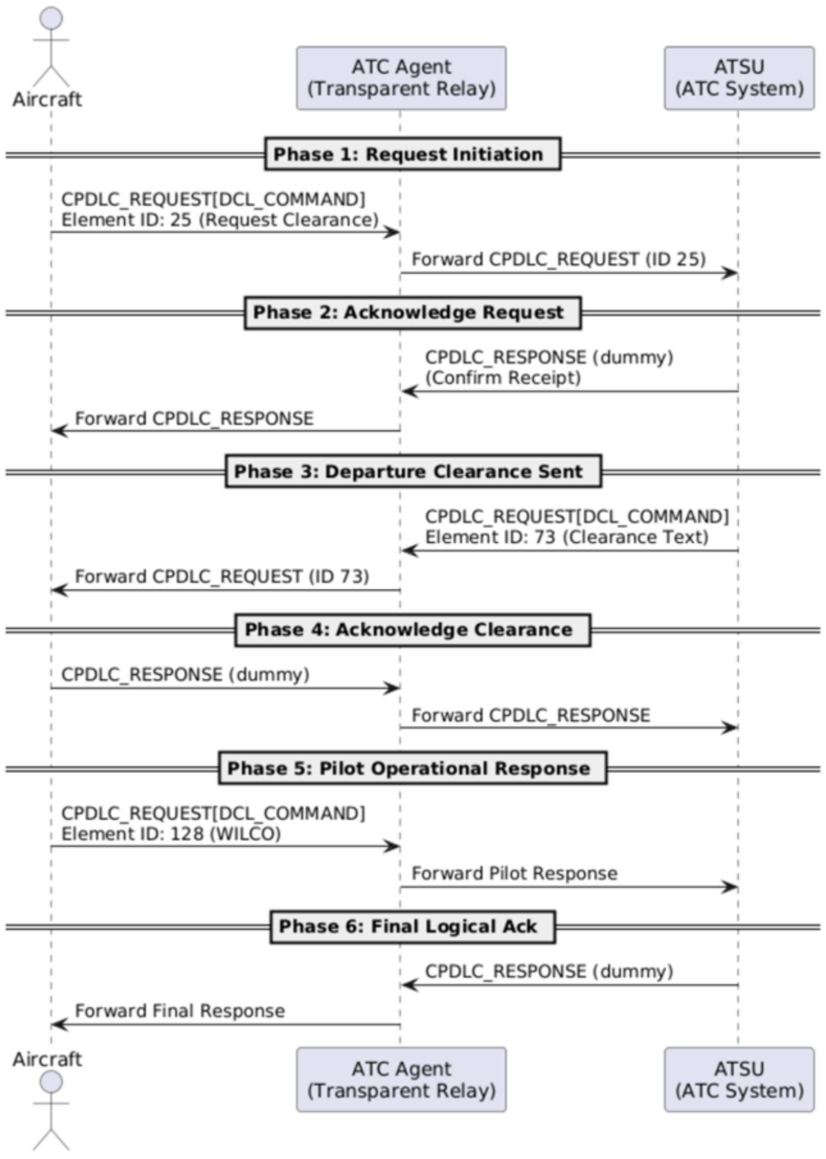

Aircraft
ATC Agent (Transparent Relay)
ATSU (ATC System)
Phase 1: Request Initiation
CPDLC_REQUEST[DCL_COMMAND] Element ID: 25 (Request Clearance)
Forward CPDLC_REQUEST (ID 25)
Phase 2: Acknowledge Request
CPDLC_RESPONSE (dummy) (Confirm Receipt)
Forward CPDLC_RESPONSE
Phase 3: Departure Clearance Sent
CPDLC_REQUEST[DCL_COMMAND] Element ID: 73 (Clearance Text)
Forward CPDLC_REQUEST (ID 73)
Phase 4: Acknowledge Clearance
CPDLC_RESPONSE (dummy)
Forward CPDLC_RESPONSE
Phase 5: Pilot Operational Response
CPDLC_REQUEST[DCL_COMMAND] Element ID: 128 (WILCO)
Forward Pilot Response
Phase 6: Final Logical Ack
CPDLC_RESPONSE (dummy)
Forward Final Response
Aircraft
ATC Agent (Transparent Relay)
ATSU (ATC System)

Figure 3.13: Departure Clearance Workflow - Strict Pair Mode (UDP)

The Departure Clearance workflow (Figure 3.13) under the strict request–response model required by UDP-based communication, ensuring that every CPDLC message exchange is explicitly acknowledged.

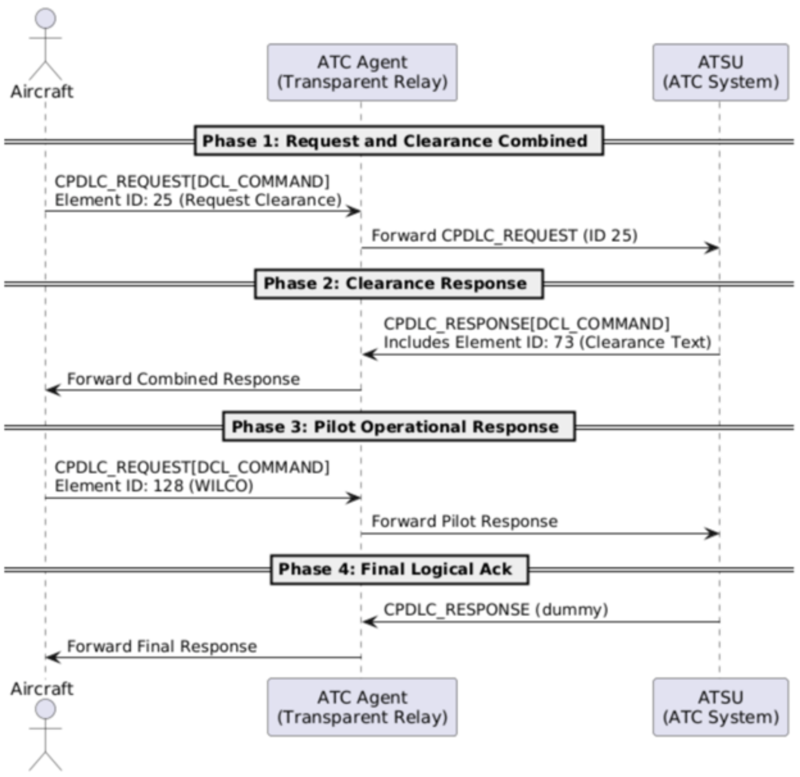

Aircraft
ATC Agent (Transparent Relay)
ATSU (ATC System)
Phase 1: Request and Clearance Combined
CPDLC_REQUEST[DCL_COMMAND] Element ID: 25 (Request Clearance)
Forward CPDLC_REQUEST (ID 25)
Phase 2: Clearance Response
CPDLC_RESPONSE[DCL_COMMAND] Includes Element ID: 73 (Clearance Text)
Forward Combined Response
Phase 3: Pilot Operational Response
CPDLC_REQUEST[DCL_COMMAND] Element ID: 128 (WILCO)
Forward Pilot Response
Phase 4: Final Logical Ack
CPDLC_RESPONSE (dummy)
Forward Final Response

Figure 3.14: Departure Clearance Workflow - Hybrid Mode (TCP)

The optimised Departure Clearance workflow (Figure 3.14) for TCP environments, where acknowledgement and payload may be combined in a single response to reduce message overhead.

3.6.1.2 D-Startup Clearance Service

In addition to this standard functionality, ATMACA extends the DCL framework to support a set of custom pre-departure services—namely D-STARTUP.

The D-STARTUP Clearance service enables pilots to request startup approval from air traffic control prior to engine ignition and pushback. Implemented as a custom extension under the DCL_COMMAND structure, this service facilitates structured, traceable coordination of ground operations during the initial pre-departure phase. The pilot sends a DCL_COMMAND with a dedicated element ID representing a startup request, and the controller responds with either a startup clearance or an appropriate status message such as WILCO, UNABLE, or STANDBY. This service ensures standardised messaging even in non-voice environments and adheres to ATMACA’s request–response model, supporting both UDP (Strict Pair Mode) and TCP (Hybrid Mode) deployments.

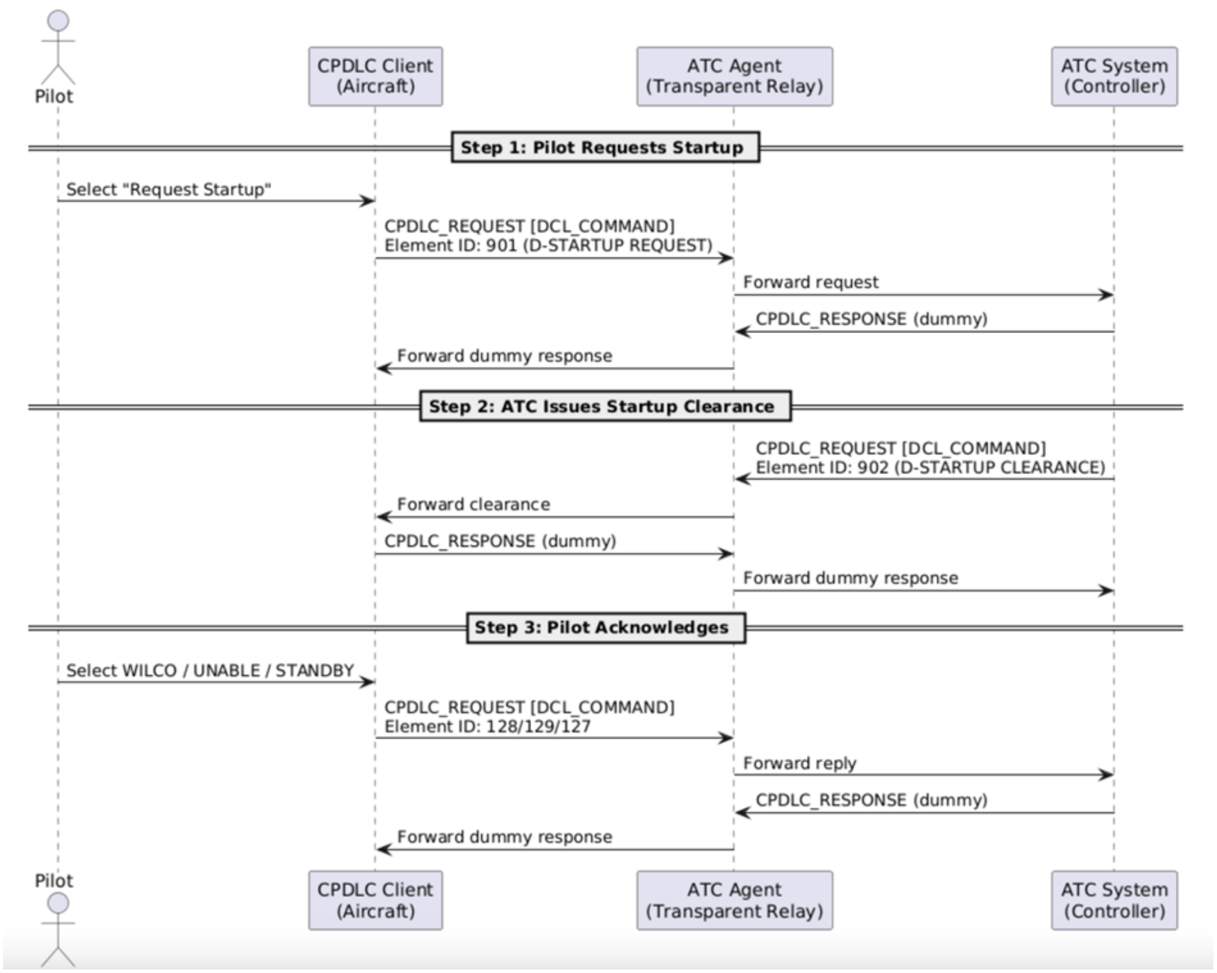

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Requests Startup
Select "Request Startup"
CPDLC_REQUEST [DCL_COMMAND] Element ID: 901 (D-STARTUP REQUEST)
Forward request
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 2: ATC Issues Startup Clearance
CPDLC_REQUEST [DCL_COMMAND] Element ID: 902 (D-STARTUP CLEARANCE)
Forward clearance
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 3: Pilot Acknowledges
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DCL_COMMAND] Element ID: 128/129/127
Forward reply
CPDLC_RESPONSE (dummy)
Forward dummy response

Figure 3.15: D-Startup Clearance Workflow - Strict Pair Mode (UDP)

Sequence diagram illustrating the D-STARTUP workflow over UDP (Figure 3.15), using DCL_COMMAND with strict request–response pairs for each message to ensure reliable delivery and acknowledgement.

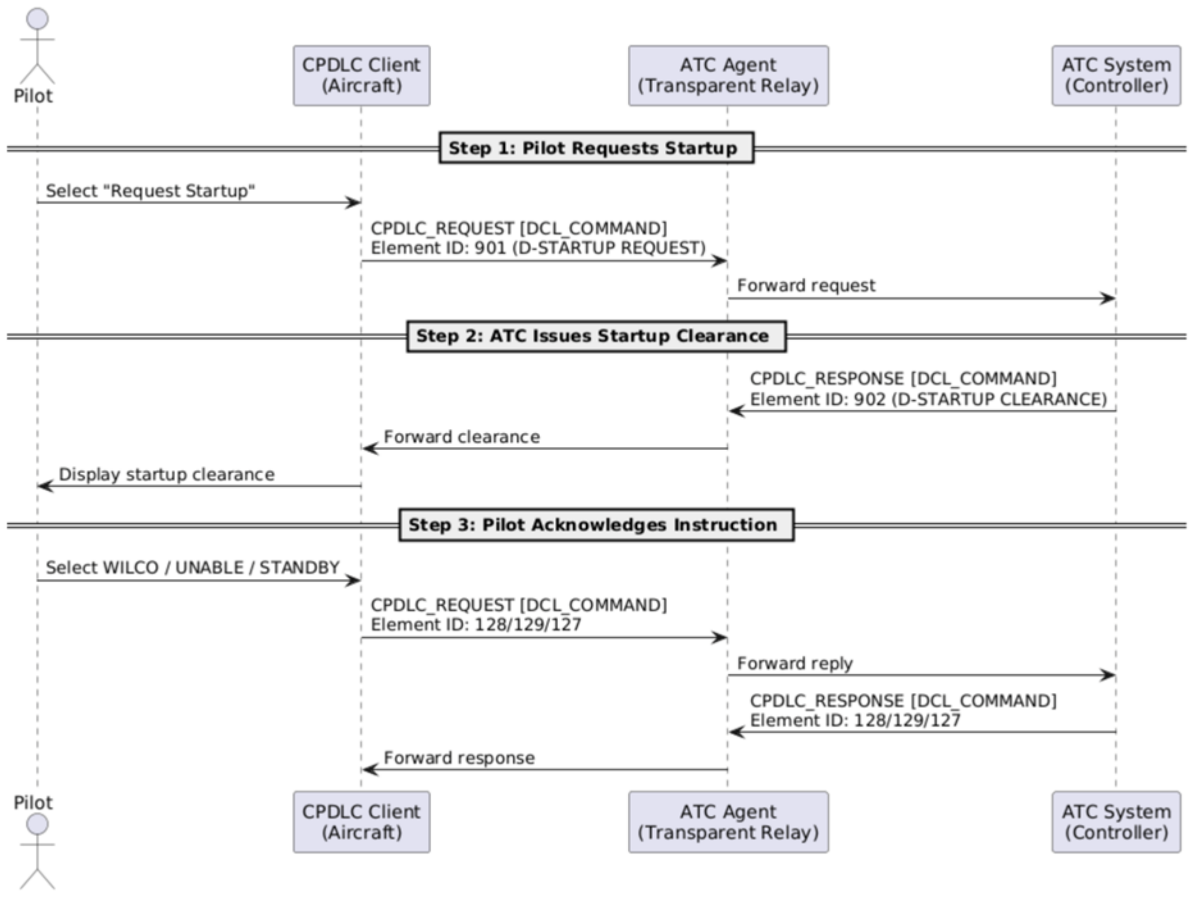

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Requests Startup
Select "Request Startup"
CPDLC_REQUEST [DCL_COMMAND] Element ID: 901 (D-STARTUP REQUEST)
Forward request
Step 2: ATC Issues Startup Clearance
CPDLC_RESPONSE [DCL_COMMAND] Element ID: 902 (D-STARTUP CLEARANCE)
Forward clearance
Display startup clearance
Step 3: Pilot Acknowledges Instruction
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DCL_COMMAND] Element ID: 128/129/127
Forward reply
CPDLC_RESPONSE [DCL_COMMAND] Element ID: 128/129/127
Forward response

Figure 3.16: Departure Clearance Workflow - Hybrid Mode (TCP)

Sequence diagram showing the D-STARTUP workflow over TCP (

Figure 3.16), streamlining communication through reliable transport without dummy acknowledgements.

3.6.1.3 D-Pushback Clearance Service

The D-PUSHBACK Clearance service supports digital coordination of aircraft pushback operations between pilots and ground controllers during the pre-departure phase. This service is implemented as a custom message type under the DCL_COMMAND structure within ATMACA and follows the same structured request–response model used across CPDLC-based workflows. The pilot initiates the exchange with a pushback request (DCL_COMMAND with a designated element ID), and the controller responds with either a pushback clearance or a status message such as WILCO, UNABLE, or STANDBY. This structured messaging flow enhances clarity, traceability, and operational safety during ramp movement, and is fully compatible with both UDP and TCP transport configurations.

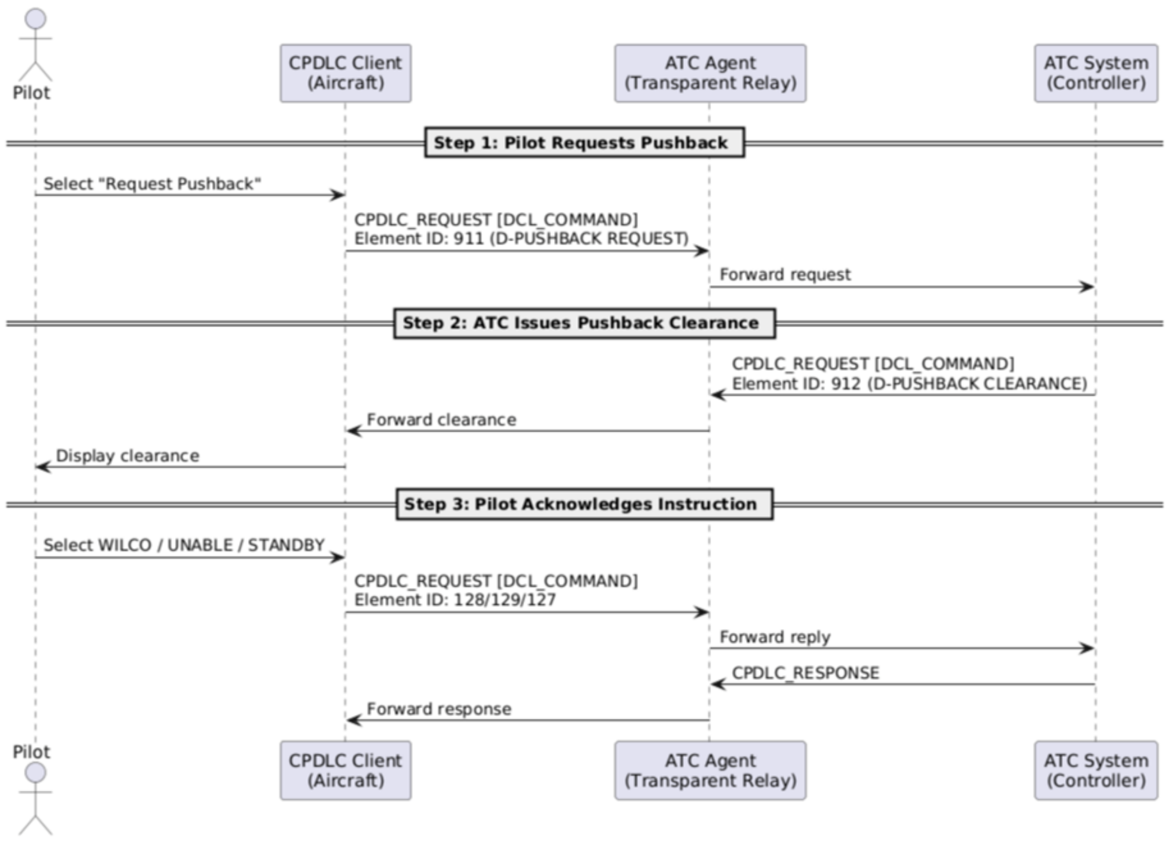

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Requests Pushback
Select "Request Pushback"
CPDLC_REQUEST [DCL_COMMAND] Element ID: 911 (D-PUSHBACK REQUEST)
Forward request
Step 2: ATC Issues Pushback Clearance
CPDLC_REQUEST [DCL_COMMAND] Element ID: 912 (D-PUSHBACK CLEARANCE)
Forward clearance
Display clearance
Step 3: Pilot Acknowledges Instruction
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DCL_COMMAND] Element ID: 128/129/127
Forward reply
CPDLC_RESPONSE
Forward response

Figure 3.17: D-Pushback Workflow - Hybrid Mode (TCP)

Sequence diagram illustrating the D-PUSHBACK clearance workflow over TCP (

Figure 3.17), reducing protocol overhead by leveraging reliable transport for streamlined exchanges.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

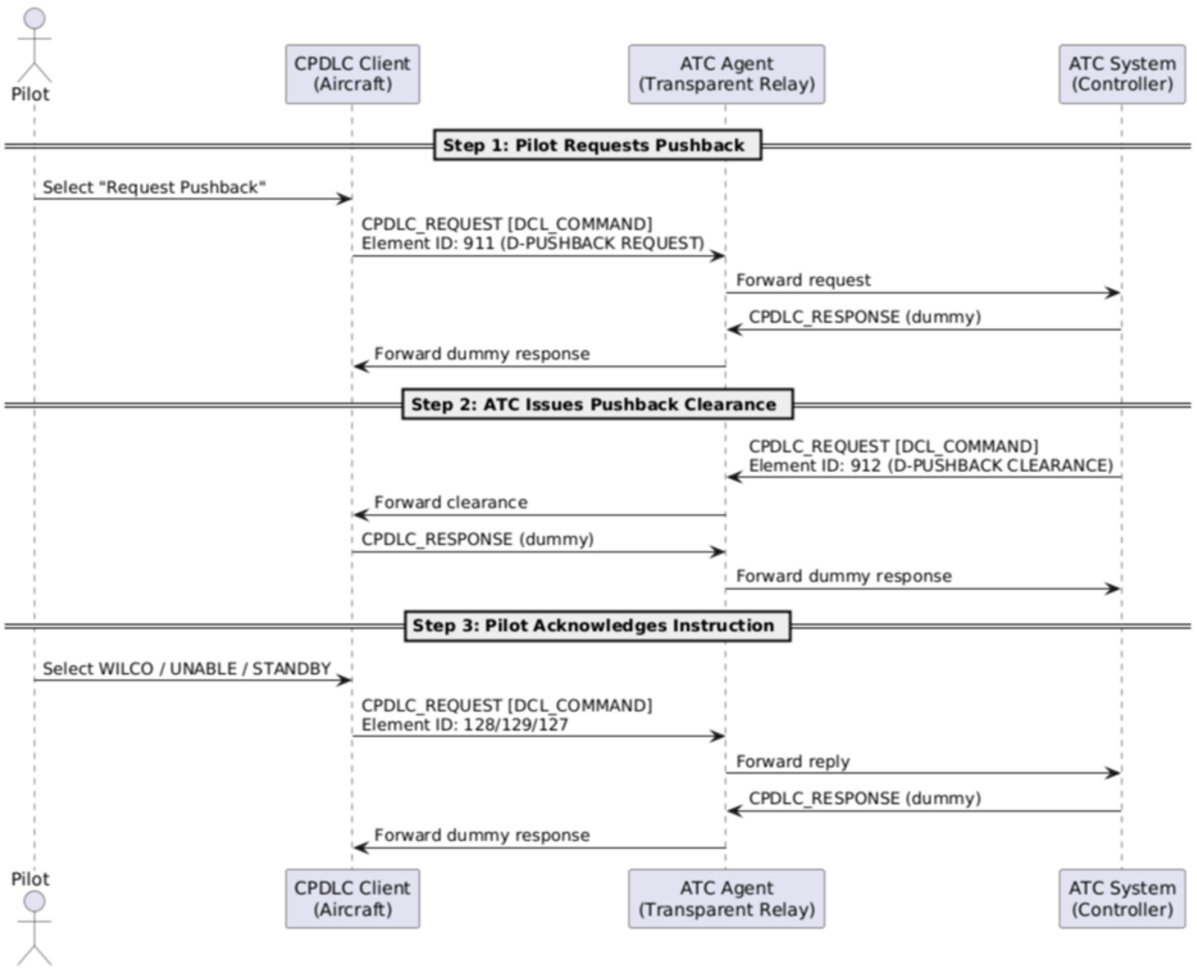

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Requests Pushback
Select "Request Pushback"
CPDLC_REQUEST [DCL_COMMAND] Element ID: 911 (D-PUSHBACK REQUEST)
Forward request
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 2: ATC Issues Pushback Clearance
CPDLC_REQUEST [DCL_COMMAND] Element ID: 912 (D-PUSHBACK CLEARANCE)
Forward clearance
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 3: Pilot Acknowledges Instruction
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DCL_COMMAND] Element ID: 128/129/127
Forward reply
CPDLC_RESPONSE (dummy)
Forward dummy response

Figure 3.18: D-Pushback Clearance Workflow - Strict Pair Mode (UDP)

Sequence diagram showing the D-PUSHBACK clearance workflow over UDP (

Figure 3.18), using DCL_COMMAND messages with strict request–response pairing to ensure complete transactional delivery.

3.6.1.4 D-Pushback Clearance Service

The D-TAXI Clearance service provides structured datalink communication for taxi instructions between the controller and pilot during surface movement operations. As with D-STARTUP and D-PUSHBACK, this service is implemented under the DCL_COMMAND structure using dedicated element IDs. The pilot can request taxi instructions from the controller, who replies with a D-TAXI clearance, possibly including taxi route details or hold instructions. The workflow supports both UDP and TCP modes, following ATMACA's request–response messaging model for reliable and traceable exchanges. D-TAXI enhances situational clarity and reduces voice frequency load during ground operations by formalizing the taxi clearance process over CPDLC.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

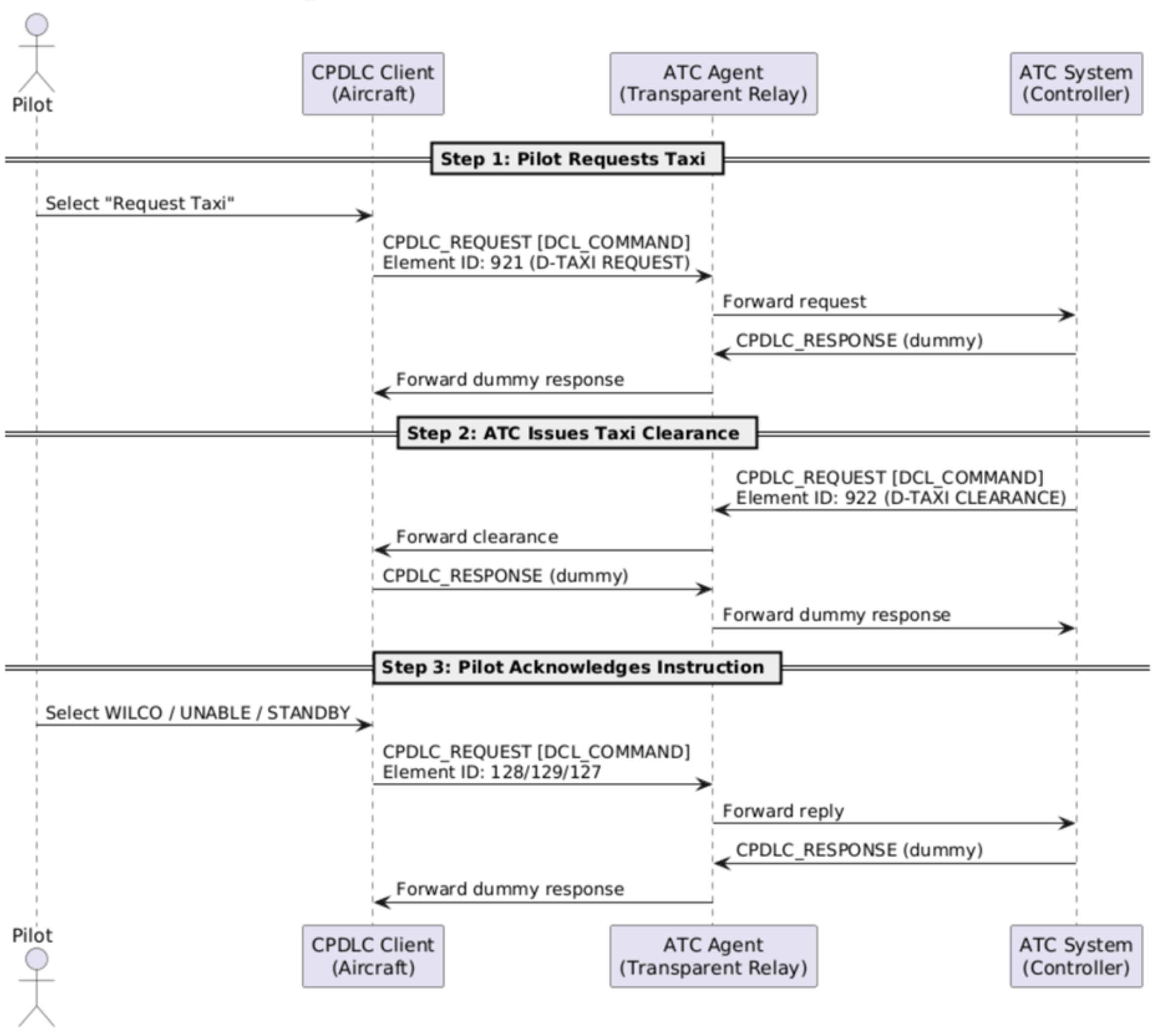

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Requests Taxi
Select "Request Taxi"
CPDLC_REQUEST [DCL_COMMAND] Element ID: 921 (D-TAXI REQUEST)
Forward request
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 2: ATC Issues Taxi Clearance
CPDLC_REQUEST [DCL_COMMAND] Element ID: 922 (D-TAXI CLEARANCE)
Forward clearance
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 3: Pilot Acknowledges Instruction
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DCL_COMMAND] Element ID: 128/129/127
Forward reply
CPDLC_RESPONSE (dummy)
Forward dummy response
Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)

Figure 3.19: D-Taxi Clearance Workflow - Strict Pair Mode (UDP)

Sequence diagram illustrating the D-TAXI workflow over UDP (

Figure 3.19), where every CPDLC message and pilot acknowledgement is exchanged using strict request–response pairs under the DCL_COMMAND structure to ensure delivery integrity.

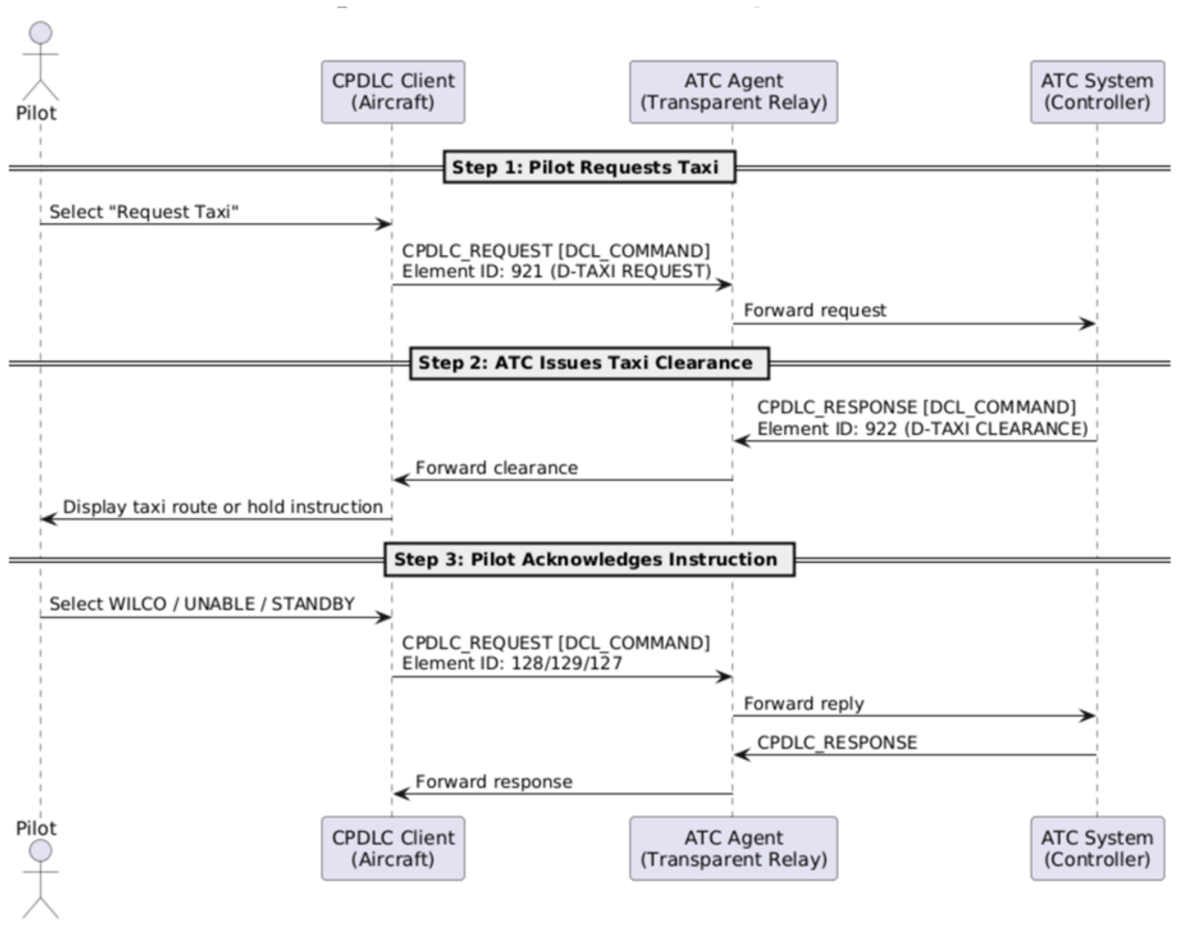

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Requests Taxi
Select "Request Taxi"
CPDLC_REQUEST [DCL_COMMAND] Element ID: 921 (D-TAXI REQUEST)
Forward request
Step 2: ATC Issues Taxi Clearance
CPDLC_RESPONSE [DCL_COMMAND] Element ID: 922 (D-TAXI CLEARANCE)
Forward clearance
Display taxi route or hold instruction
Step 3: Pilot Acknowledges Instruction
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DCL_COMMAND] Element ID: 128/129/127
Forward reply
CPDLC_RESPONSE
Forward response

Figure 3.20: D-Taxi Clearance Workflow - Hybrid Mode (TCP)

Sequence diagram showing the D-TAXI workflow over TCP (

Figure 3.20), using a reliable transport layer while preserving logical request–response cycles for clearance issuance and acknowledgement

3.6.1.5 ATC Communication Management (ACM)

The ATC Communication Management (ACM) service plays a central role in supporting the Transfer of Data Authority within CPDLC operations. Through ACM messages such as CONTACT and MONITOR, the current ATSU instructs the pilot to initiate communication with the next ATSU, either via voice or logon. These exchanges serve as operational triggers for the formal data authority transition, often complemented by the use of DAT_AUTH_DESIG messages. As a result, ACM forms the communication backbone for seamless handover between ATC units, ensuring that both voice and datalinks are reassigned in a structured, ICAO-compliant manner.

3.6.1.5.1 Transfer of Data Authority Without Ground-Ground Connectivity (T-ATSU Initiated)

The following UML sequence diagram illustrates the workflow for transferring data authority and voice communication in scenarios without supporting ground-ground connectivity, where the **T-ATSU initiates** the transfer by instructing the aircraft to contact the next data authority (NDA), with acknowledgements and link establishment handled entirely over the air-ground CPDLC communication path.

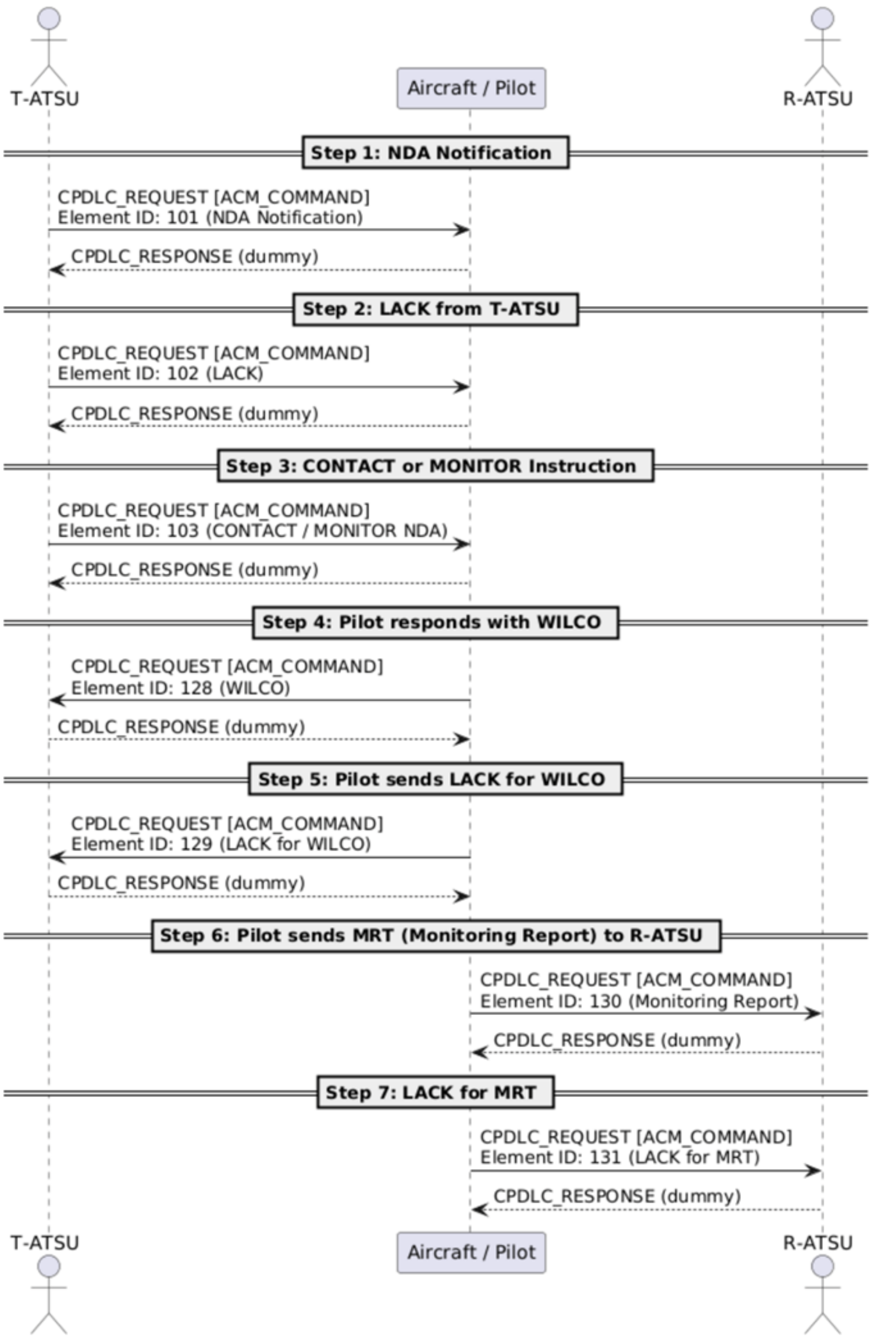

T-ATSU
Aircraft / Pilot
R-ATSU
Step 1: NDA Notification
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 101 (NDA Notification)
CPDLC_RESPONSE (dummy)
Step 2: LACK from T-ATSU
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 102 (LACK)
CPDLC_RESPONSE (dummy)
Step 3: CONTACT or MONITOR Instruction
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 103 (CONTACT / MONITOR NDA)
CPDLC_RESPONSE (dummy)
Step 4: Pilot responds with WILCO
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 128 (WILCO)
CPDLC_RESPONSE (dummy)
Step 5: Pilot sends LACK for WILCO
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 129 (LACK for WILCO)
CPDLC_RESPONSE (dummy)
Step 6: Pilot sends MRT (Monitoring Report) to R-ATSU
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 130 (Monitoring Report)
CPDLC_RESPONSE (dummy)
Step 7: LACK for MRT
CPDLC_REQUEST [ACM_COMMAND]
Element ID: 131 (LACK for MRT)
CPDLC_RESPONSE (dummy)
T-ATSU
Aircraft / Pilot
R-ATSU

Figure 3.21: Transfer of Data Authority – T-ATSU Initiated (Strict Mode – UDP)

The Transfer of Data Authority workflow in Strict Pair Mode (UDP) (Figure 3.21), where each CPDLC request and response is explicitly acknowledged to ensure delivery integrity.

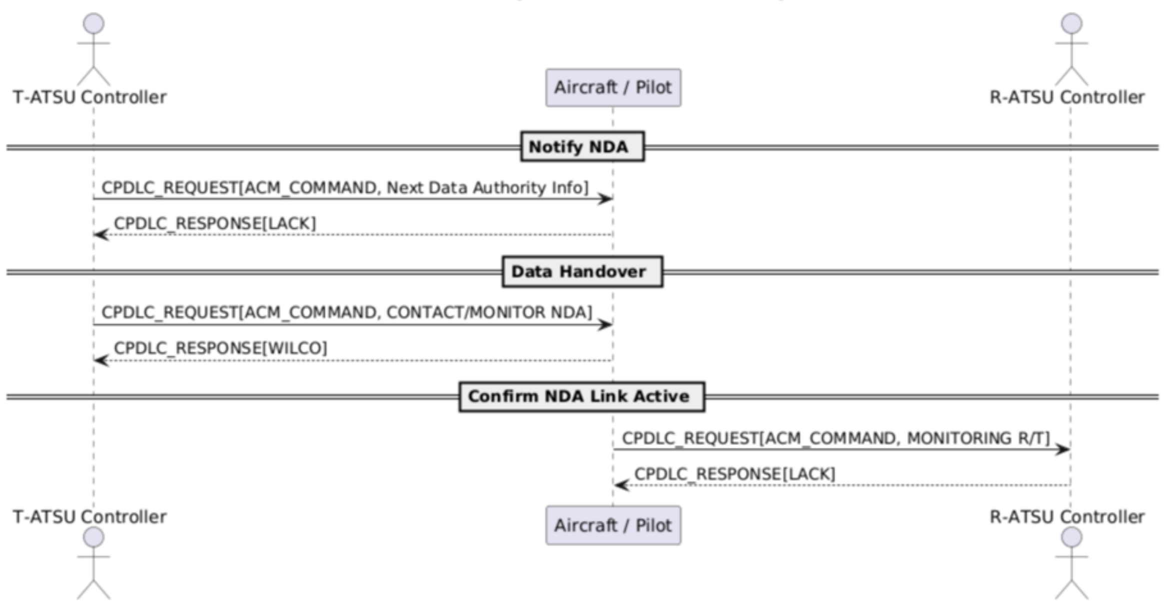


Figure 3.22: Transfer of Data Authority – T-ATSU Initiated (Hybrid Mode – TCP)

The Transfer of Data Authority workflow in Hybrid Mode (TCP), where acknowledgements may be embedded or implicit due to reliable transport (

Figure 3.22).

3.6.1.6 ATC Clearance (ACL)

All ATC instructions and clearances exchanged via the ATMACA CPDLC implementation are encapsulated using the ACL_COMMAND wrapper. Each clearance message corresponds to a specific CPDLC element identified by its standardised element ID, and is sent from the ground system (ATC) to the aircraft. Depending on the response attributes defined for the element, the aircraft may respond with a mandatory operational reply such as WILCO, UNABLE, or STANDBY, or may remain silent if no response is required. In UDP-based deployments, ATMACA enforces a strict request–response model where every ACL_COMMAND request must be paired with a response — even when operational content is not required — to ensure protocol compliance. In TCP-based deployments, responses may be embedded or skipped if permitted by the transport-layer reliability and element behaviour. This dual-mode structure ensures flexible, standards-compliant ACL handling across all communication environments.

3.6.1.6.1 UPLINK Clearance/Instruction

This section describes the workflow for uplink CPDLC clearance and instruction messages initiated by the ATC, such as route modifications, altitude changes, or speed adjustments. These messages are transmitted as ACL_COMMAND requests and are typically followed by pilot responses including WILCO, UNABLE, or STANDBY. In UDP-based deployments (

Figure 3.23**)**, each message and response is encapsulated in strict request–response pairs to ensure delivery confirmation, whereas TCP-based workflows (

Figure 3.24**)** allow streamlined exchanges with inline acknowledgements.

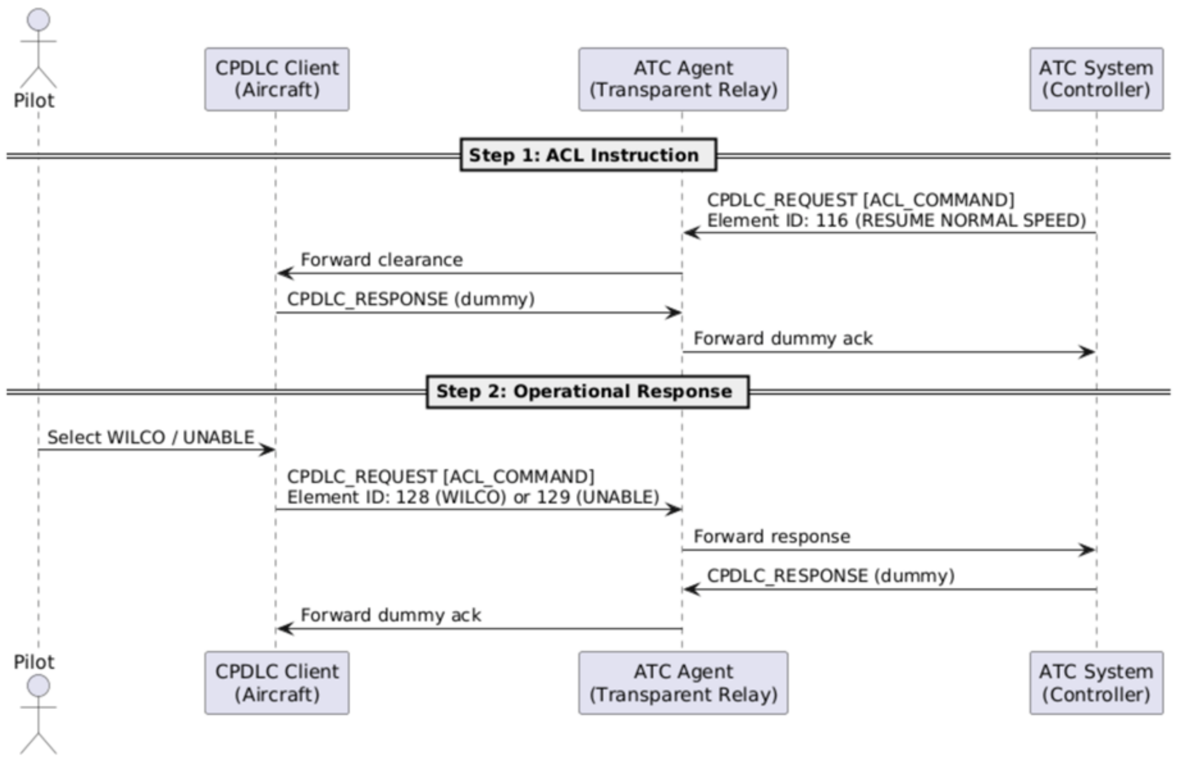


Figure 3.23: Uplink – RESUME NORMAL SPEED (Strict Mode – UDP)

Uplink instructions are reliably conveyed, confirmed, and acted upon across the ATMACA communication chain.

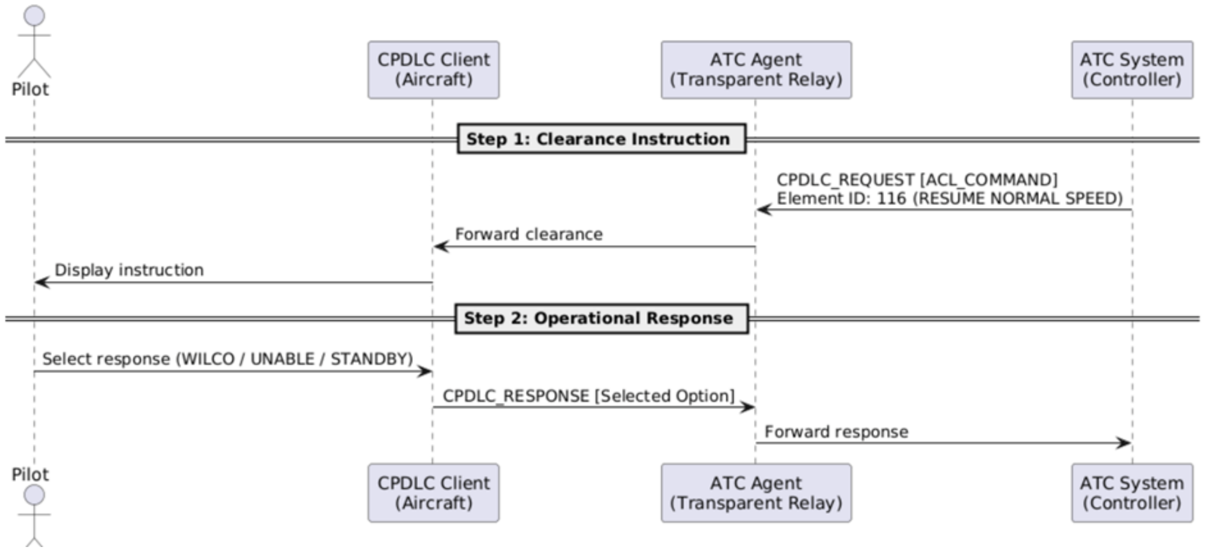

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Clearance Instruction
CPDLC_REQUEST [ACL_COMMAND] Element ID: 116 (RESUME NORMAL SPEED)
Forward clearance
Display instruction
Step 2: Operational Response
Select response (WILCO / UNABLE / STANDBY)
CPDLC_RESPONSE [Selected Option]
Forward response

Figure 3.24: Uplink – RESUME NORMAL SPEED (Hybrid Mode – TCP)

3.6.1.6.2 DOWNLINK Requests

This section outlines the structure and communication flow for downlink CPDLC request messages initiated by the flight crew. These messages are typically sent to request clearances, route changes, altitude amendments, or specific operational permissions. Each request is encapsulated in an ACL_COMMAND message and transmitted via the aircraft's CPDLC client through the ATC Agent to the ground ATC system. Depending on the transport mode, UDP deployments (

Figure 3.26**)** require strict request–response pairs to confirm message delivery, while TCP workflows (

Figure 3.25**)** may use embedded acknowledgements. The following diagrams illustrate how these downlink requests are processed, relayed, and answered within the ATMACA protocol architecture.

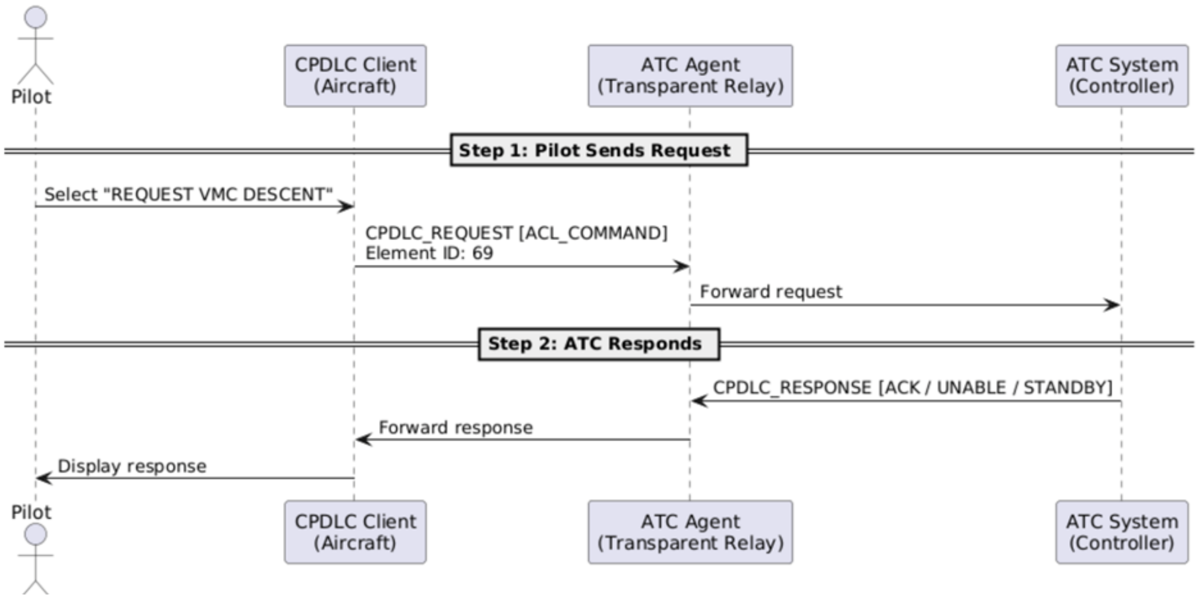

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Sends Request
Select "REQUEST VMC DESCENT"
CPDLC_REQUEST [ACL_COMMAND] Element ID: 69
Forward request
Step 2: ATC Responds
CPDLC_RESPONSE [ACK / UNABLE / STANDBY]
Forward response
Display response
Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)

Figure 3.25: Downlink – REQUEST VMC DESCENT (Hybrid Mode – TCP)

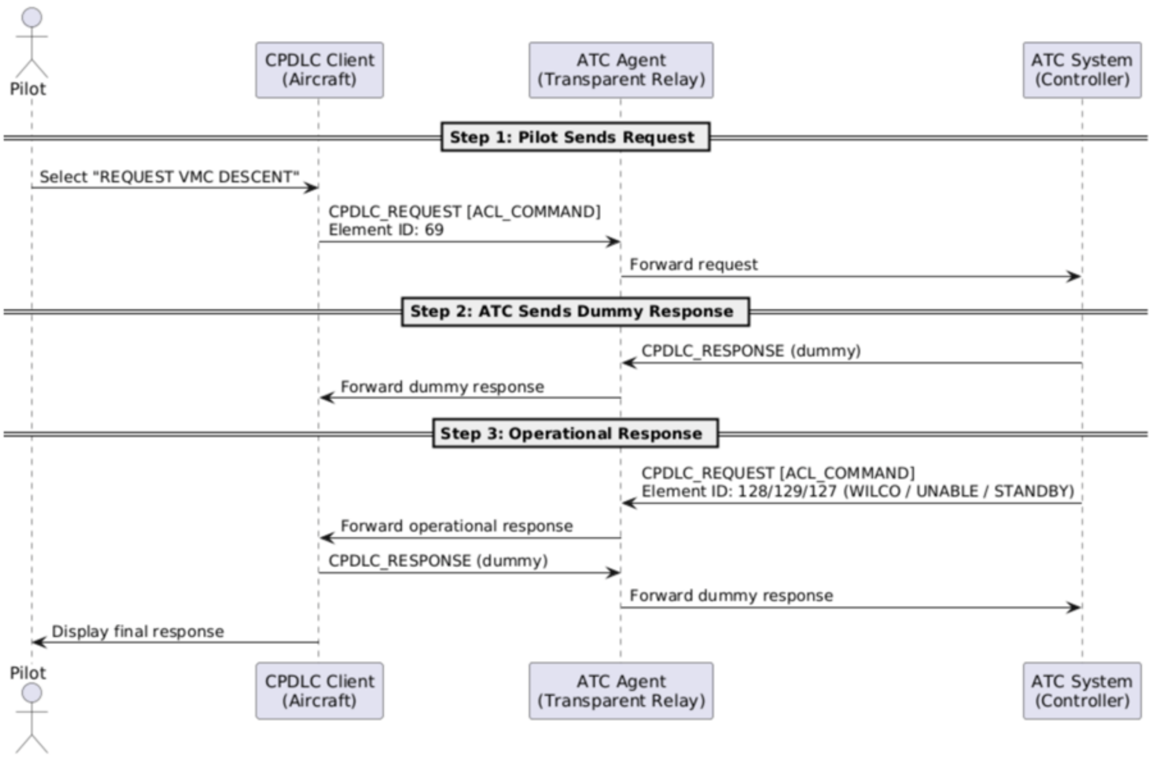

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
ATC System (Controller)
Step 1: Pilot Sends Request
Select "REQUEST VMC DESCENT"
CPDLC_REQUEST [ACL_COMMAND] Element ID: 69
Forward request
Step 2: ATC Sends Dummy Response
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 3: Operational Response
CPDLC_REQUEST [ACL_COMMAND] Element ID: 128/129/127 (WILCO / UNABLE / STANDBY)
Forward operational response
CPDLC_RESPONSE (dummy)
Forward dummy response
Display final response

Figure 3.26: Downlink – REQUEST VMC DESCENT (Strict Pair – UDP)

3.6.1.7 Downstream Clearance (DSC)

The Downstream Clearance (DSC) service allows pilots to request and receive clearances from a downstream Air Traffic Services Unit (D-ATSU) while still under the control of their current ATSU. This facilitates early coordination for route continuation, boundary crossing, or arrival procedures. Within the ATMACA protocol, DSC exchanges are encapsulated using DSC_COMMAND messages and adhere to either a Strict Pair Mode (UDP) (

Figure 3.27) or a Hybrid Mode (TCP) **(**

Figure 3.28**)**, depending on the transport layer. Each workflow ensures message integrity, acknowledgement handling, and accurate routing through the ATC Agent. The following diagrams illustrate the complete DSC message sequence, from pilot request to clearance delivery and operational response.

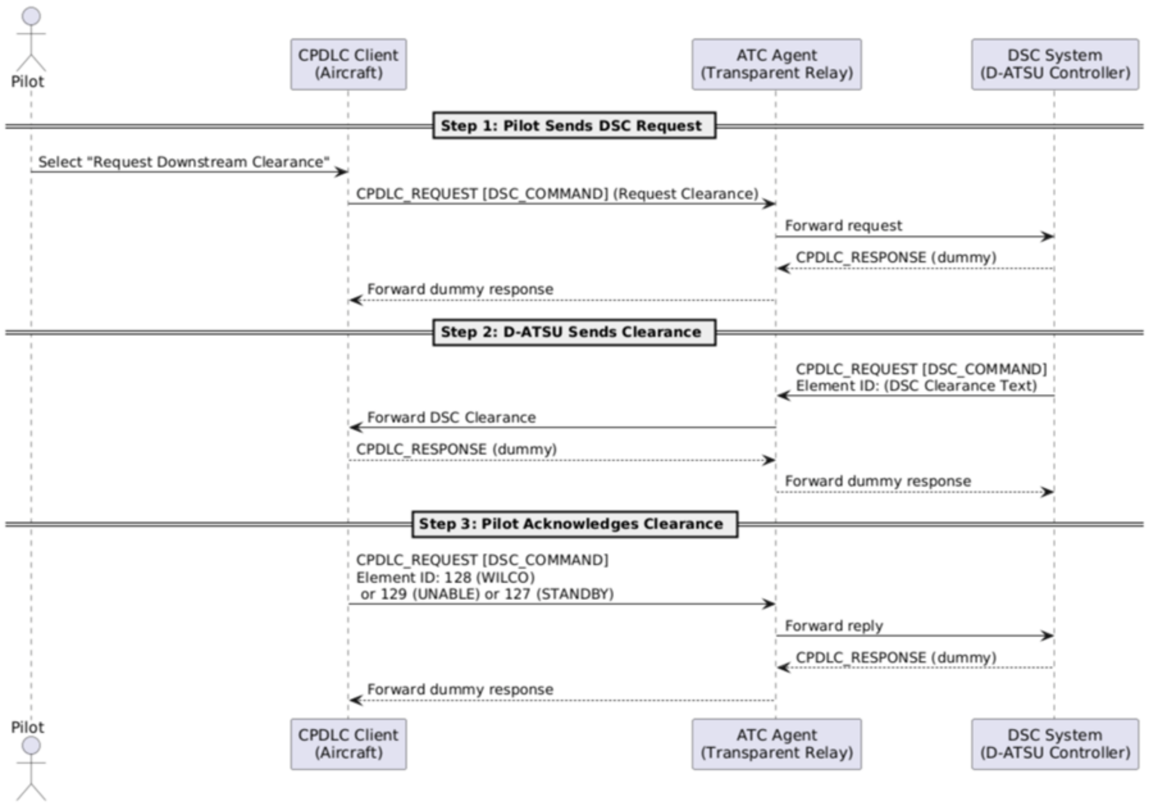

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
DSC System (D-ATSU Controller)
Step 1: Pilot Sends DSC Request
Select "Request Downstream Clearance"
CPDLC_REQUEST [DSC_COMMAND] (Request Clearance)
Forward request
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 2: D-ATSU Sends Clearance
CPDLC_REQUEST [DSC_COMMAND] Element ID: (DSC Clearance Text)
Forward DSC Clearance
CPDLC_RESPONSE (dummy)
Forward dummy response
Step 3: Pilot Acknowledges Clearance
CPDLC_REQUEST [DSC_COMMAND] Element ID: 128 (WILCO) or 129 (UNABLE) or 127 (STANDBY)
Forward reply
CPDLC_RESPONSE (dummy)
Forward dummy response

Figure 3.27: Downstream Clearance Workflow (UDP - Strict Pair Mode)

The DSC workflow over UDP, where each CPDLC message and acknowledgment is exchanged as a request–response pair to ensure delivery reliability.

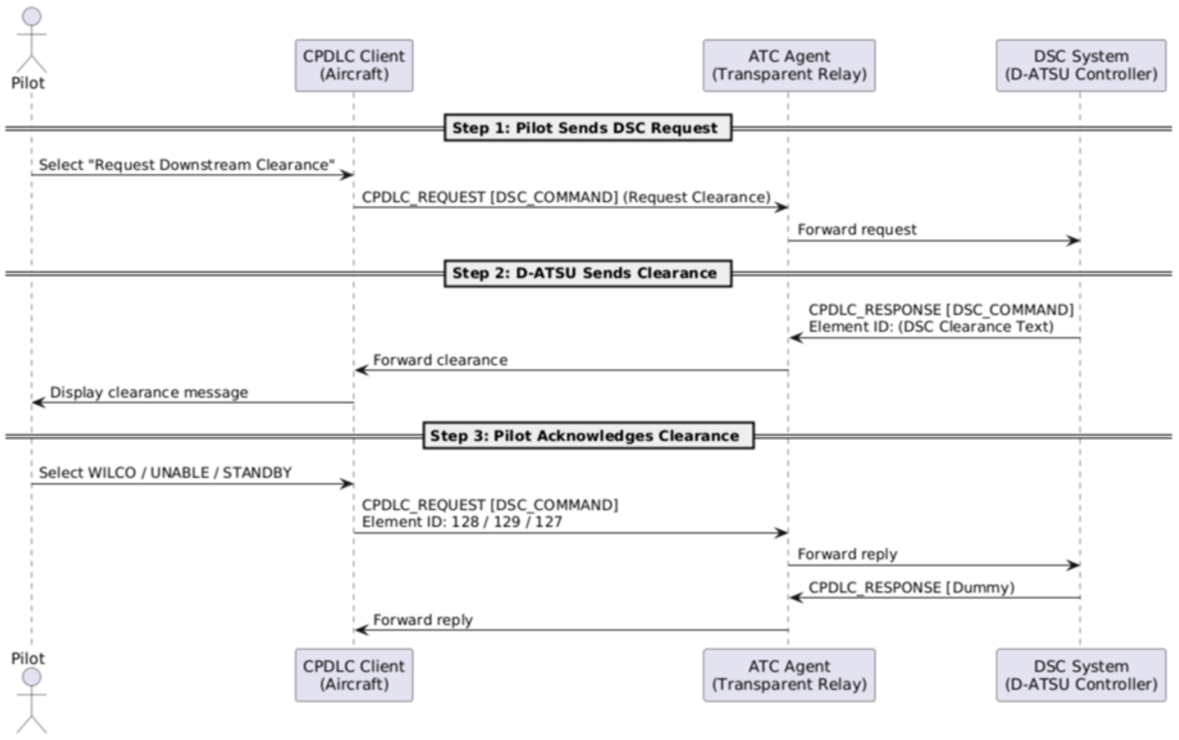

Pilot
CPDLC Client (Aircraft)
ATC Agent (Transparent Relay)
DSC System (D-ATSU Controller)
Step 1: Pilot Sends DSC Request
Select "Request Downstream Clearance"
CPDLC_REQUEST [DSC_COMMAND] (Request Clearance)
Forward request
Step 2: D-ATSU Sends Clearance
CPDLC_RESPONSE [DSC_COMMAND] Element ID: (DSC Clearance Text)
Forward clearance
Display clearance message
Step 3: Pilot Acknowledges Clearance
Select WILCO / UNABLE / STANDBY
CPDLC_REQUEST [DSC_COMMAND] Element ID: 128 / 129 / 127
Forward reply
CPDLC_RESPONSE [Dummy]
Forward reply

Figure 3.28: Downstream Clearance Workflow (TCP - Hybrid Mode)

The DSC workflow over TCP, enabling a more efficient exchange without dummy responses, while preserving the logical sequence of request, clearance, and reply.

3.7 CPDLC Interactions within ATMACA

3.7.1 Interactions with ATMACA protocol Stack and Stack Interface

The ATMACA Protocol Stack is a modular and extensible communication framework built to support high-performance, connection-oriented or datagram-based messaging between distributed systems. Inspired by concepts from Diameter, Message Queuing Telemetry Transport (MQTT), and RADIUS, it facilitates message transmission, peer discovery, state machine orchestration, and robust transaction handling. At the heart of this stack are peer entities—ATMACAPeer—which manage connections and handle the transmission of protocol messages through TCP or UDP transports.

The ATMACA protocol stack allows developers to access core protocol functionalities. These include Messaging and Transaction Control (Figure 3.29) features that allow request-response message interchange. In the case of CPDLC all of this is implemented by DLCM and its DLIC module. So, CPDLC sends messages through DLCM, not communicating directly to the protocol stack.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

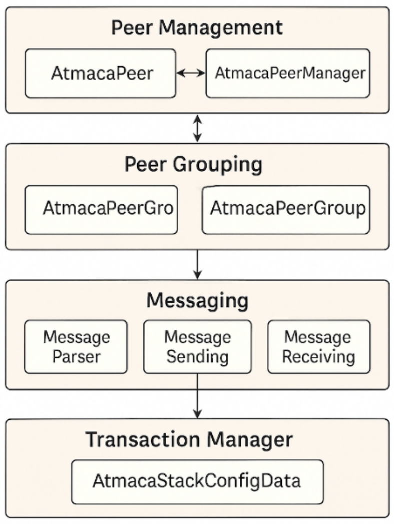


Figure 3.29: ATMACA Messaging and Peer Management Architecture

Another key component is the ATMACA Protocol Stack Interface, which forms a vital extension of the core ATMACA Protocol Stack, introducing layers and mechanisms tailored for advanced routing, peer management, message observation, and application-specific behaviour (Figure 3.30). This layer enhances the flexibility, observability, and control of peer-to-peer interactions, especially in dynamic air traffic management environments where resilience and adaptability are crucial.

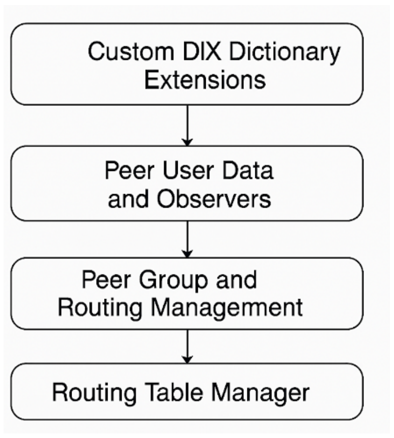

Custom DIX Dictionary Extensions
Peer User Data and Observers
Peer Group and Routing Management
Routing Table Manager

Figure 3.30: ATMACA Routing Management Hierarchy

CPDLC will interact with some of these features, especially those related to the DIX Dictionary Extensions, as these are crucial to define the application's implemented message in the base protocol. Other features related to peers and routing needed will be handled by DLCM or the protocol itself.

As a summary, to be implemented on the framework's protocol, the CPDLC is expected to:

- Interact with the Custom DIX dictionary Extensions to implement all needed CPDLC messages into the base protocol. To do so:
    - DlcBaseDIXDictionary will be expanded with additional CPDLC elements to handle all possible data. Each AVP will be registered using the most suitable DIX type.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

  - CPDLC is not expected to modify the DlcDIXDictionary AVPs.

- CPDLC is not expected to interact directly with the protocol stack or with the stack's interface, routing and peer features. This will be managed for CPDLC by DLCM.

3.7.2 Interactions with DLCM

DLCM functionalities are key for CPDLC service provision in the ATMACA framework. Through DLCM context, connection, session and mobility are managed (Figure 3.31. These allow to deliver reliable, flexible and context-aware communications.

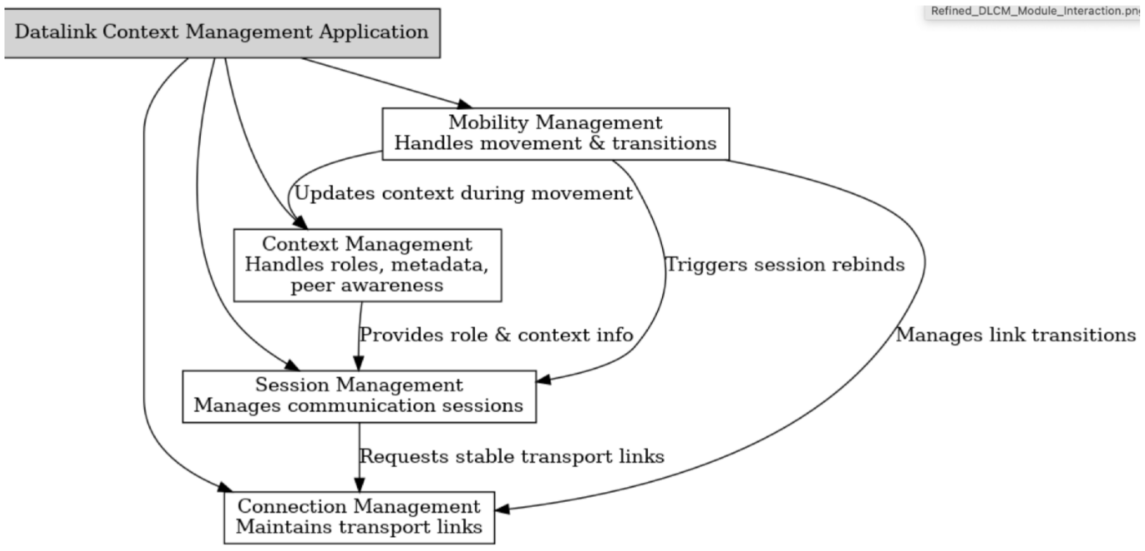

Refined_DLCM_Module_Interaction.png
Datalink Context Management Application
Mobility Management
Handles movement & transitions
Updates context during movement
Context Management
Handles roles, metadata, peer awareness
Triggers session rebinds
Provides role & context info
Manages link transitions
Session Management
Manages communication sessions
Requests stable transport links
Connection Management
Maintains transport links

Figure 3.31: DLCM Module Interaction

CPDLC will use the following management features:

- Context (operational metadata, roles and connectivity)
- Session (application-level)
- Connection (L4 links)

CPDLC will not directly use Mobility functionalities

3.7.2.1 Context Management Module

The Context Management Module keeps track of user roles, system state, and operational assignments, helping coordinate services across different nodes. It is responsible for establishing and maintaining an operational context for each node (client or server) in the network. It enables seamless session and role continuity across mobility events and network transitions. Its role is crucial for handover execution.

The Context Management architecture (

Figure 3.32) relies on several key components that work together to support accurate role assignment, operational awareness, and seamless coordination across the network. These include:

- Context Entity – Represents a logical unit of communication, such as a user session or operational role, uniquely identified and tracked across the system.
- Context Registry – Maintains an authoritative list of active contexts, including associated metadata like node ID, realm, and assigned role.
- Role Binding Table – Maps each context entity to one or more operational roles (e.g., Controlling, Monitoring), ensuring correct role-based access and coordination.
- Peer Presence Tracker – Keeps real-time awareness of peer availability and status, supporting transitions, handovers, and redundancy logic.
- Context Sync Interface – Facilitates context updates, replication, and failover handling between distributed nodes.
- Lifecycle Manager – Oversees context creation, updates, termination, and cleanup processes, ensuring consistency throughout dynamic events.

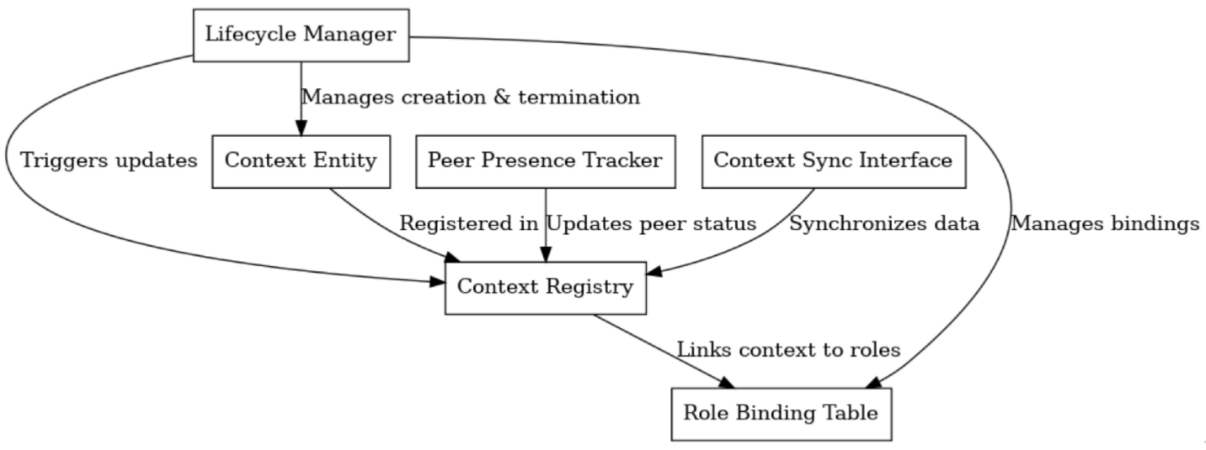


Figure 3.32: Context Management Architectural Elements Diagram

Context's main structure is summarised in the following table (Table 3.2):

Table 3.2: Context Main Structure

Field Name	Type	Description
context_id	string	Unique identifier for the context (e.g., delivery@saw.tr.atm, THY1AB@thy.tr.atm).
context_name	string	Human-readable name for the context (e.g., "Delivery Control").
context_type	string (Enum)	Identifies the type of context (e.g., ATC, Flight).
context_status	string (Enum)	Identifies the **current operational status** (please refer to context status table)
context_connectivity	object	Contains **node role assignments** (e.g., Controlling, Mirroring, and Monitoring Nodes).
context_routing	object	Contains **routing information** such as atc_agent_addr, gateways, etc.
context_data	object	Contains **session data** and **adjacent contexts**.
context_metadata	object	Describes context creation, updates, and additional metadata.
context_history	array of objects	Tracks key context changes for **audit**, **troubleshooting**, and **recovery**.

CPDLC is expected to:

- Access Context information to be used for message automation. This context information will be accessed through:
    - CONTEXT_PULL_REQUEST, to retrieve the latest context definition from the CM Agent.
- Receive updates through CONTEXT_UPDATE_NOTIFY_REQUEST.
- Receive role updates through CONTEXT_ROLE_CHANGE_NOTIFY_REQUEST

3.7.2.2 Session Management Module

The Session Management Module (

Figure 3.33 is in charge of starting, maintaining, and ending communication sessions, ensuring that data flows smoothly even during transitions. Each session represents a logical, stateful communication link between two endpoints: a context owner (e.g., a controller's context) and a remote peer (e.g., an aircraft or another ground system). Sessions are dynamically managed and can be handed over from one context to another to support operational continuity across sector changes, controller shifts, or facility handovers.

To maintain robust, persistent communication across dynamic network environments, the Session Management architecture includes a set of core components that work together to manage session lifecycle, association, and recovery. These elements form the backbone of ATMACA's session control model:

- Session Entity – Represents a unique, persistent communication instance between endpoints, identified by a session ID and used throughout the session lifecycle.
- Session Table – Serves as the active repository for tracking all live sessions, maintaining metadata such as session status, associated nodes, and timestamps.

- Session Binder – Handles the creation, lookup, and binding of application logic to session entities, ensuring that application flows are correctly tied to session context.
- Transport Association Registry – Maps sessions to one or more underlying transport connections, allowing the protocol to manage multiplexing and recover connections during failovers.
- Timeout & Expiry Manager – Monitors session activity and enforces timeout rules, handling expiration, cleanup, and idle session termination to maintain system efficiency.
- Recovery & Handover Controller – Manages the recovery of sessions during failovers or mobility events by re-binding session state and restoring communication continuity.

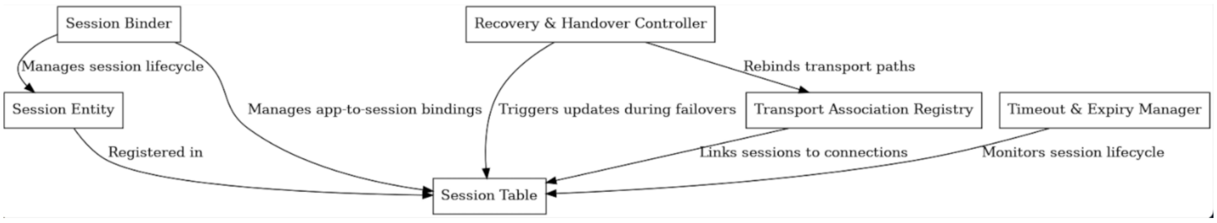
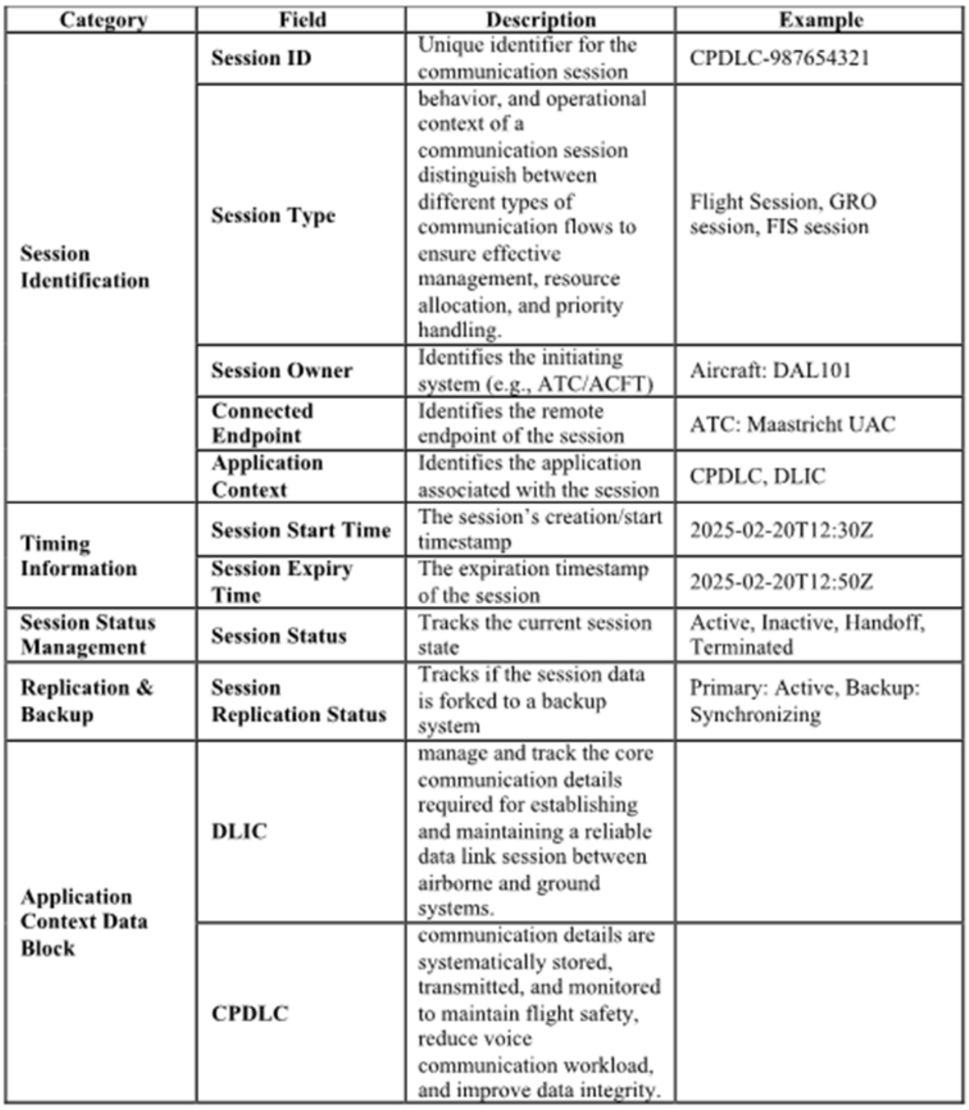

Session Binder
Recovery & Handover Controller
Manages session lifecycle
Rebinds transport paths
Session Entity
Manages app-to-session bindings
Triggers updates during failovers
Transport Association Registry
Timeout & Expiry Manager
Registered in
Links sessions to connections
Monitors session lifecycle
Session Table

Figure 3.33: Session Management Architectural Elements Diagram

Table 3.3 represents the session data structure **Error! Reference source not found.**:

Table 3.3: Session Data Structure



Category	Field	Description	Example
Session Identification	Session ID	Unique identifier for the communication session	CPDLC-987654321
	Session Type	behavior, and operational context of a communication session distinguish between different types of communication flows to ensure effective management, resource allocation, and priority handling.	Flight Session, GRO session, FIS session
	Session Owner	Identifies the initiating system (e.g., ATC/ACFT)	Aircraft: DAL101
	Connected Endpoint	Identifies the remote endpoint of the session	ATC: Maastricht UAC
	Application Context	Identifies the application associated with the session	CPDLC, DLIC
Timing Information	Session Start Time	The session's creation/start timestamp	2025-02-20T12:30Z
	Session Expiry Time	The expiration timestamp of the session	2025-02-20T12:50Z
Session Status Management	Session Status	Tracks the current session state	Active, Inactive, Handoff, Terminated
Replication & Backup	Session Replication Status	Tracks if the session data is forked to a backup system	Primary: Active, Backup: Synchronizing
Application Context Data Block	DLIC	manage and track the core communication details required for establishing and maintaining a reliable data link session between airborne and ground systems.	
	CPDLC	communication details are systematically stored, transmitted, and monitored to maintain flight safety, reduce voice communication workload, and improve data integrity.	

CPDLC is expected to:

- Retrieve all useful session information with application-specific payloads using SESSION_DATA_PULL_REQUEST.
- Summit application session data updates using SESSION_DATA_PUSH_REQUEST.
- Retrieve new session data after a SESSION_OWNER_CHANGE_NOTIFY_REQUEST is received.

3.7.2.3 Connection Management Module

The Connection Management Module (

Figure 3.34) handles the network links themselves, checking their health, recovering from failures, and keeping everything stable. It is responsible for managing the operational state and lifecycle of communication links between mobile clients (such as flight deck applications) and ground-based agents (such as CM and ATC Agents). It supports DLIC lifecycle messages for initiating and terminating operational presence.

The Connection Management module ensures that reliable, low-level communication paths remain active and responsive across dynamic network environments. Its architecture is composed of several interrelated components that maintain transport-layer continuity and enable seamless connection recovery. These elements are detailed below:

- Connection Entity – Represents a single transport-level connection between nodes, carrying metadata such as protocol type, address, and connection state.
- Connection Table – Acts as the central registry for all active connections, tracking their state, associated nodes, and connection parameters in real time.
- Transport Multiplexer – Manages multiple parallel transport channels, enabling load balancing, prioritization, and multi-homing across network interfaces.
- Heartbeat Manager – Periodically sends heartbeat messages and monitors acknowledgments to detect connection health and trigger alerts on failure.
- Failover Controller – Coordinates recovery actions in the event of link failure, including re-routing, session re-binding, or switching to alternative transport paths.
- Transport Pool – Maintains a catalogue of available transport resources (e.g., TCP, UDP, SCTP), allowing the system to establish new connections dynamically as needed.

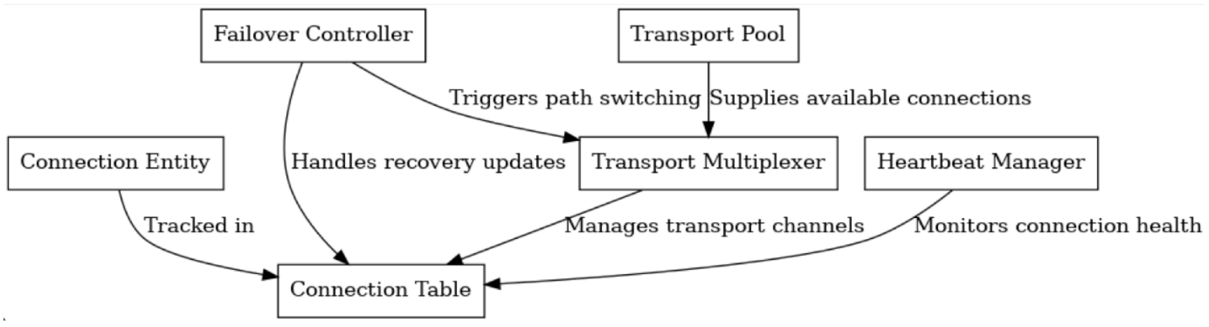
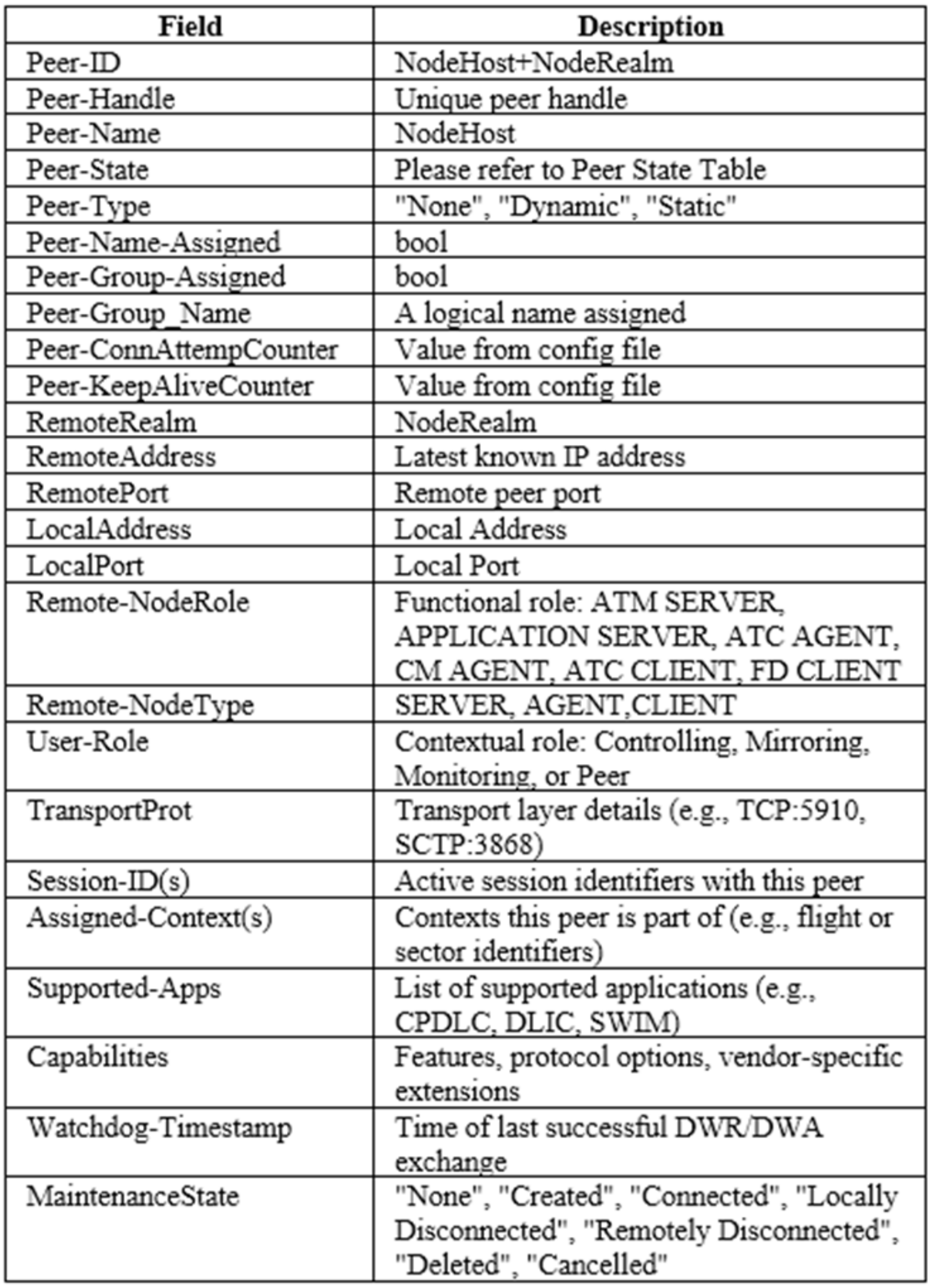


Figure 3.34: Connection Management Architectural Elements Diagram

The mobility management module uses the following routing and peer tables shown in Table 3.44:

Table 3.4: Peer Table Structure



Field	Description
Peer-ID	NodeHost+NodeRealm
Peer-Handle	Unique peer handle
Peer-Name	NodeHost
Peer-State	Please refer to Peer State Table
Peer-Type	"None", "Dynamic", "Static"
Peer-Name-Assigned	bool
Peer-Group-Assigned	bool
Peer-Group_Name	A logical name assigned
Peer-ConnAttempCounter	Value from config file
Peer-KeepAliveCounter	Value from config file
RemoteRealm	NodeRealm
RemoteAddress	Latest known IP address
RemotePort	Remote peer port
LocalAddress	Local Address
LocalPort	Local Port
Remote-NodeRole	Functional role: ATM SERVER, APPLICATION SERVER, ATC AGENT, CM AGENT, ATC CLIENT, FD CLIENT
Remote-NodeType	SERVER, AGENT,CLIENT
User-Role	Contextual role: Controlling, Mirroring, Monitoring, or Peer
TransportProt	Transport layer details (e.g., TCP:5910, SCTP:3868)
Session-ID(s)	Active session identifiers with this peer
Assigned-Context(s)	Contexts this peer is part of (e.g., flight or sector identifiers)
Supported-Apps	List of supported applications (e.g., CPDLC, DLIC, SWIM)
Capabilities	Features, protocol options, vendor-specific extensions
Watchdog-Timestamp	Time of last successful DWR/DWA exchange
MaintenanceState	"None", "Created", "Connected", "Locally Disconnected", "Remotely Disconnected", "Deleted", "Cancelled"

The DLIC function (Figure 3.35) enables the establishment and maintenance of operational presence for ATC clients (e.g., controller systems, aircraft) within the ATMACA architecture. This module orchestrates the logical association between client applications and their corresponding operational context by facilitating connection initiation, context updates, inter-agent handovers, and peer forwarding. It ensures seamless identification, mobility support, and system-wide synchronisation—especially across distributed CM/ATC Agents and operational sectors.

DLIC leverages structured message exchanges between ATC clients, ATC Agents, and FlightDeck clients to manage the lifecycle of operational presence and connectivity. The core capabilities include logon validation, session continuity, agent contact handovers, and support for ground message forwarding (Figure 3.365).

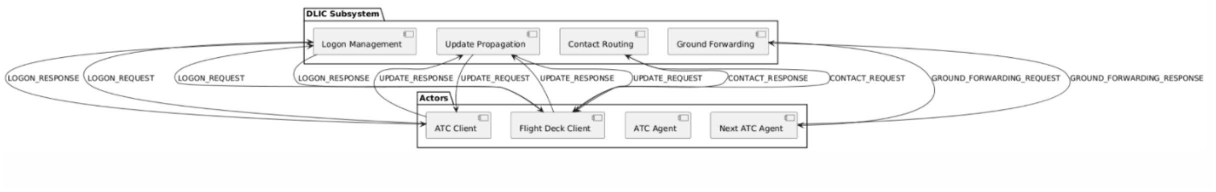


Figure 3.35: DLIC Functional Subsystem

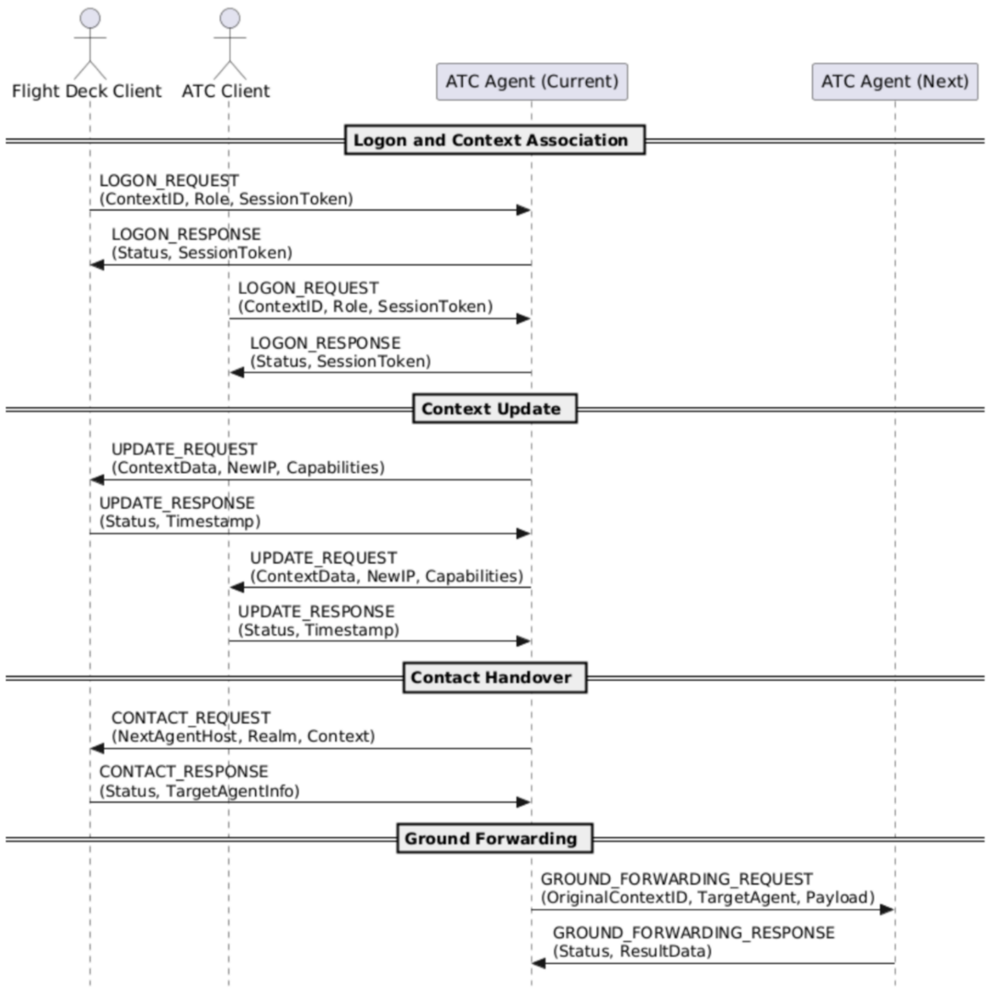

Flight Deck Client
ATC Client
ATC Agent (Current)
ATC Agent (Next)
Logon and Context Association
LOGON_REQUEST (ContextID, Role, SessionToken)
LOGON_RESPONSE (Status, SessionToken)
LOGON_REQUEST (ContextID, Role, SessionToken)
LOGON_RESPONSE (Status, SessionToken)
Context Update
UPDATE_REQUEST (ContextData, NewIP, Capabilities)
UPDATE_RESPONSE (Status, Timestamp)
UPDATE_REQUEST (ContextData, NewIP, Capabilities)
UPDATE_RESPONSE (Status, Timestamp)
Contact Handover
CONTACT_REQUEST (NextAgentHost, Realm, Context)
CONTACT_RESPONSE (Status, TargetAgentInfo)
Ground Forwarding
GROUND_FORWARDING_REQUEST (OriginalContextID, TargetAgent, Payload)
GROUND_FORWARDING_RESPONSE (Status, ResultData)

Figure 3.36: DLIC Message Flow

Through DLIC, CPDLC is expected to:

- Use LOGON_REQUEST and LOGON_RESPONSE messages to execute all needed DLIC service functions prior to other services.
- Request context updates using UPDATE_REQUEST and receiving UPDATE_RESPONSE
- Execute handover messages using ACM and CONTACT_REQUEST and CONTACT_RESPONSE.

- Forward messages via Ground using GROUND_FORWARDING_REQUEST and GROUND_FORWARDING_RESPONSE.

3.7.2.4 Mobility Management Module

The Mobility Management Module (Figure 3.377) ensures that communication continues without interruption when users or systems move across different regions or networks. This module abstracts various forms of mobility in air traffic control scenarios, mapping them to the appropriate DLCM subcomponents based on their functional responsibilities.

ATMACA recognises four distinct types of mobility as shown in Table 3.5, where each type is mapped to the DLCM module (s) covering them:

- **User Mobility** – supported by the Context Management module
- **Session Mobility** – supported by the Session Management module
- **Terminal (Station) Mobility** – supported by the Connection Management module
- **Service Mobility** – supported by the Connection Management module and Session Management Module

Table 3.5: Types of Mobility

Mobility Type	Managed By	Purpose
User Mobility	Context Management Module	Allows users to dynamically change contexts while retaining roles
Session Mobility	Session Management Module	Enables session continuity across control or sector handovers
Terminal Mobility	Connection Management Module	Maintains connectivity for mobile devices and clients
Service Mobility	Service Registry & Routing Logic	Ensures services are reachable regardless of user or server location

The Mobility Management module provides the logic and infrastructure required to maintain communication continuity as nodes move across operational domains, network boundaries, or airspace regions. The architecture is composed of several key elements, each contributing to real-time mobility tracking, context migration, and seamless session reassignment:

- **Mobility Context**: Represents the current mobility state of a node, including its last known position, active role, and network association.
- **Mobility Table:** A dynamic record of all mobile entities in the system, used to track location updates, movement history, and active bindings.
- **Handover Manager**: Coordinates role transitions and communication handoffs when nodes move between agents, facilities, or control areas.
- **Rebinding Agent:** Manages the reassignment of session, context, and transport associations during mobility events to ensure service continuity.
- **Mobility Event Trigger**: Detects movement conditions that require system-level response, such as FIR crossing, agent boundary transitions, or connectivity changes.

- **Location Monitor**: Continuously tracks node location using identifiers or beacon updates, feeding data to mobility controllers for proactive handover decisions.

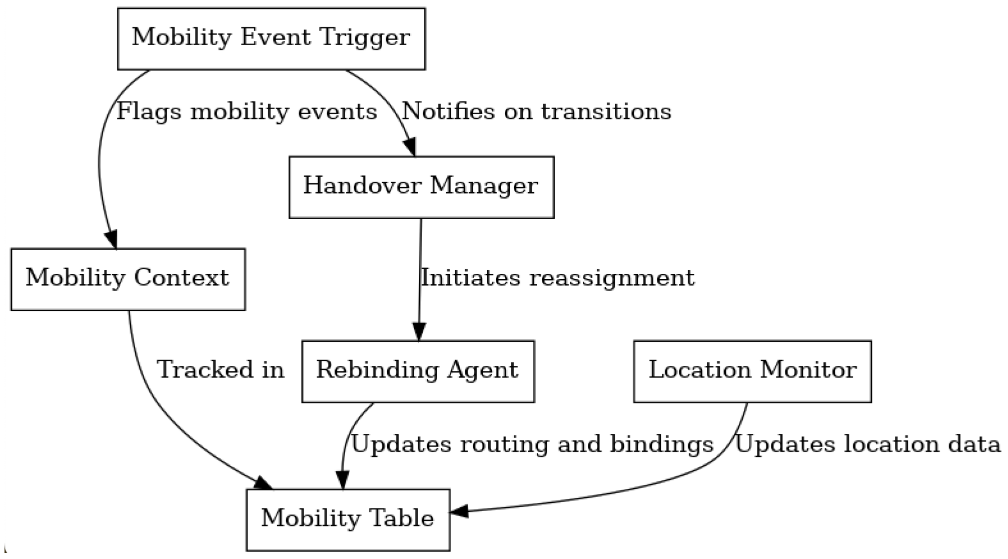


Figure 3.37: Mobility Management Architectural Elements Diagram

CPDLC is not expected to interact with Mobility Management functions. All needed capabilities will be managed by DLCM and provided to CPDLC.

3.7.3 Interactions with HMI

HMI provides a graphic interface that allows pilots and controllers to easily operate the backend CPDLC application functionalities. ATMACA implements several HMIs. Those related to ATCos and controllers present a CPDLC panel, adapted to the functions needed for each role.

Through HMI, messages can be sent using free text or predefined options, which are pre-filled with context data. Moreover, the CPDLC communications log can be accessed and displayed to the ATCo when needed. All new messages are automatically displayed. HMI also shows users the timers related to each communication.

Another key feature of HMI is automatic handover coordination. When an ATCo wants to execute transfer of a flight the HMI calls all needed CPDLC functions, sending the required messages and managing the process for the ATCo.

Figure 3.38 shows an example of an interface representing the CPDLC messaging module within the Flight Deck HMI. It allows pilots to conduct clear and structured digital communications with Air Traffic Control (ATC). The panel is split into two key sections:

- **CPDLC Message Inbox:** This area displays a chronological list of uplink and downlink CPDLC messages exchanged between the aircraft and ATC:
    - **ATC Uplink Messages** are shown in quotes, such as:

LTAA ACC → "CLIMB TO FL350"

These are instructions or responses received from ATC.

- **Expected Response Options** are shown immediately below each uplink, indicating how the pilot can acknowledge or respond:
  - [WILCO] – Will comply
  - [STANDBY] – Need more time
  - [UNABLE] – Cannot comply
  - [NEGATIVE] – Rejection (in other contexts)

- **Downlink Requests from the pilot:**

(e.g., REQUEST DIRECT TO LANKA) are shown with a timestamp and status message like Awaiting ATC Response, indicating a pending state.

- **Compose New Message Panel:**

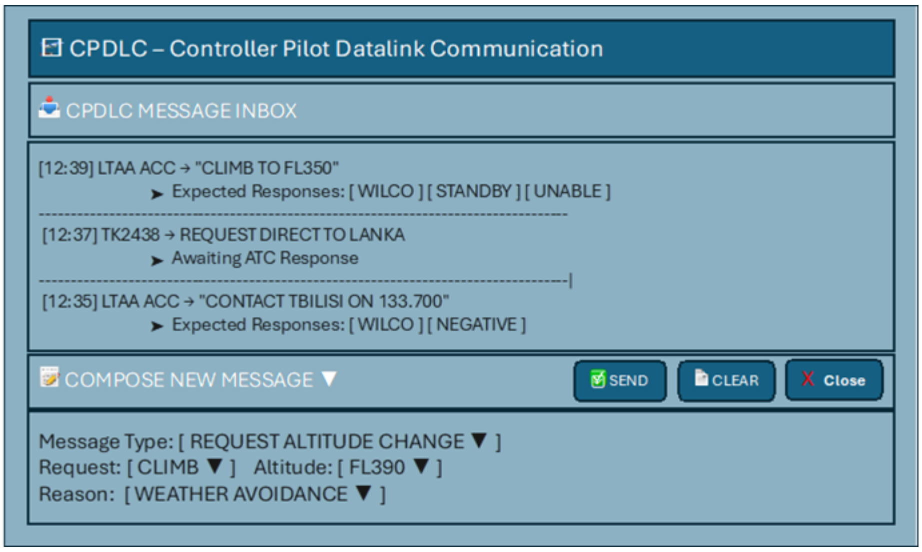
This section enables pilots to create a new downlink message using a structured format:

- **Message Type Selector:**

Allows pilots to choose the type of request, such as [REQUEST ALTITUDE CHANGE], [REQUEST DIRECT TO FIX], etc.

- **Subfields for the Request:**
  - Request: e.g., CLIMB
  - Altitude: e.g., FL390
  - Reason: e.g., WEATHER AVOIDANCE

These values are typically selected from drop-down lists to ensure compliance with CPDLC syntax and to minimize input errors.


CPDLC – Controller Pilot Datalink Communication

CPDLC MESSAGE INBOX

[12:39] LTAA ACC → "CLIMB TO FL350"
➤ Expected Responses: [WILCO] [STANDBY] [UNABLE]

[12:37] TK2438 → REQUEST DIRECT TO LANKA
➤ Awaiting ATC Response

[12:35] LTAA ACC → "CONTACT TBILISI ON 133.700"
➤ Expected Responses: [WILCO] [NEGATIVE]

COMPOSE NEW MESSAGE ▼

Message Type: [REQUEST ALTITUDE CHANGE ▼]
Request: [CLIMB ▼] Altitude: [FL390 ▼]
Reason: [WEATHER AVOIDANCE ▼]

Figure 3.38: Example of a CPDLC ATM Flight Deck HMI

To provide these functionalities HMI is expected to:

- Interact with CPDLC services and its functions to allow users to compose and send messages based on the role and associated context information in the “Compose new message panel”. This also includes coordinating the required handover messages upon execution of automatic handover.
- Access the CPDLC message logger to display all previously recorded message information to the final user in the “message inbox”.

Interact with additional features if requested by the HMI developers in future discussions.

4 CPDLC MESSAGE DEFINITION

4.1 ATMACA Message Format and Structure

ATMACA defines a binary message format (Figure 4.1) which consists of an ATMACA-header and concatenated information-elements called DIX (Datalink Information eXchange), each consisting of header and data parts.

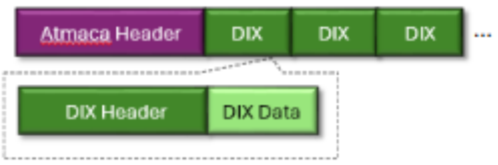


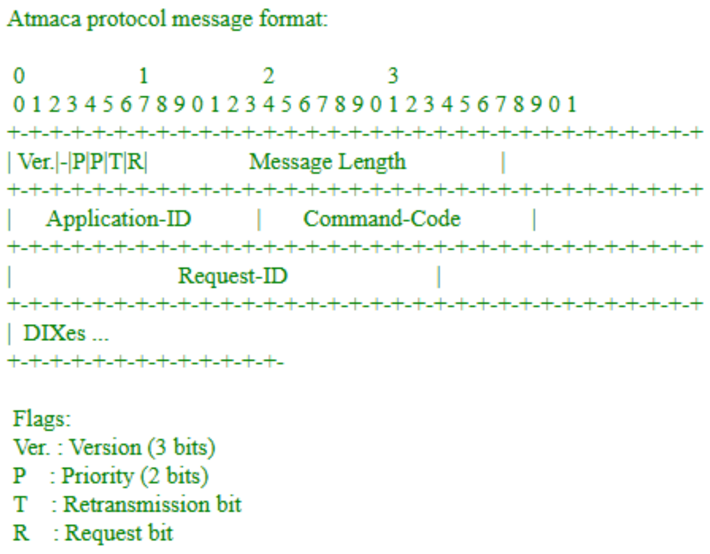
Figure 4.1: ATMACA Message format

ATMACA Header = { Flags, Length, Code, AppId, Request-Id }

DIX Header = { Flags, Code, Length, Vendor-Id (Opt) }

4.1.1 ATMACA Message Structure

ATMACA messages follow the structure shown in Figure 4.2.



```
Atmaca protocol message format:

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ver.|-|P|P|T|R|            Message Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Application-ID         |       Command-Code          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Request-ID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  DIXes ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-

 Flags:
 Ver. : Version (3 bits)
 P    : Priority (2 bits)
 T    : Retransmission bit
 R    : Request bit
```

Figure 4.2: ATMACA Message Structure

The first octet is called "flags".

The current "version" is considered as '0'

The "priority"-bits (P) determines a priority to the message

The "request" bit (R) represents if the message is a request message or a response message.

Both the request and the response for a given command share the same Command-Code. R-bit is used to distinguish this situation.

The “retransmission” bit (T) is to be used to indicate retransmitted request messages

Message-Length: Indicates the length of the ATMACA message including the header fields and the padded DIXes. Such that, the Message Length is always a multiple of 4.

Command-Code: Used to communicate the command associated with the message. So, it determines the content of the message.

Application-ID: Identifies which application the message is applicable. The application can be predefined ATMACA application, or a vendor-specific application.

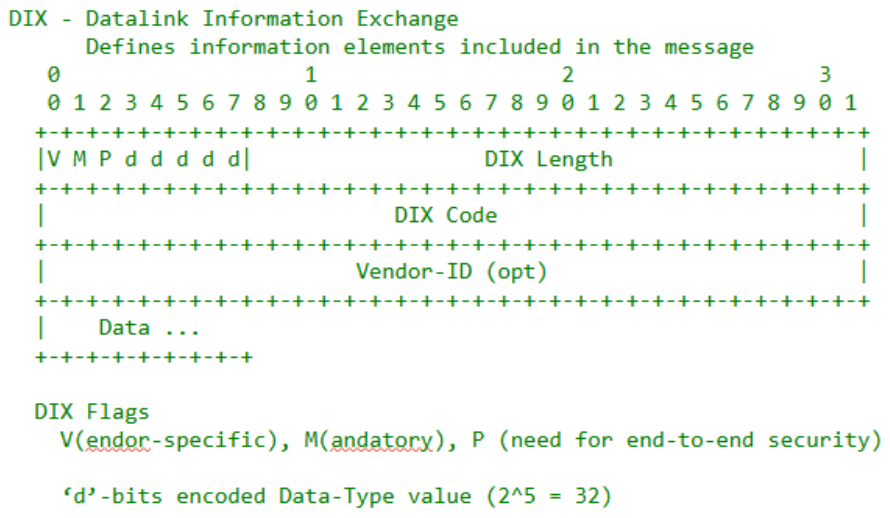
In general, the combination of Command-Code and Application-ID determines the format of the message completely.

Request-ID: Identifies the request message of transaction-based messaging, such that both request and corresponding response shall consist of the same identifier.

4.1.2 ATMACA DIX Structure

ATMACA DIXs follow the structure shown in

Figure 4.3.



```
DIX - Datalink Information Exchange
      Defines information elements included in the message
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |V M P d d d d d|                 DIX Length                    |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                           DIX Code                            |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                        Vendor-ID (opt)                        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |    Data ...
  +-+-+-+-+-+-+-+-+

  DIX Flags
    V(endor-specific), M(andatory), P (need for end-to-end security)

    ‘d’-bits encoded Data-Type value (2^5 = 32)
```

Figure 4.3: ATMACA DIX Structure

Flags:

The ‘V’ bit (Vendor Specific) indicates whether the optional Vendor-ID field is present in the DIX header. When set, the DIX Code belongs to the specific vendor code address space.

The ‘M’ bit (Mandatory) indicates whether the receiver of the DIX MUST parse and understand the semantics of the DIX, including its content.

The receiving entity MUST return an appropriate error message if it receives a DIX that has the M-bit set but does not understand it.

DIXes with the 'M' bit cleared are informational only; a receiver that receives a message with such a DIX that is not supported, or whose value is not supported, MAY simply ignore the DIX.

The five d-bits are considered to include the datatype of the DIX with predefined enumerated values

DIX Length: Indicates the number of octets in this DIX including the DIX Code field, DIX Length field, DIX Flags field, Vendor-ID field (if present), and the DIX Data field. If a message is received with an invalid attribute length, the message MUST be rejected.

DIX Code: Combined with the Vendor-Id field, identifies the attribute uniquely.

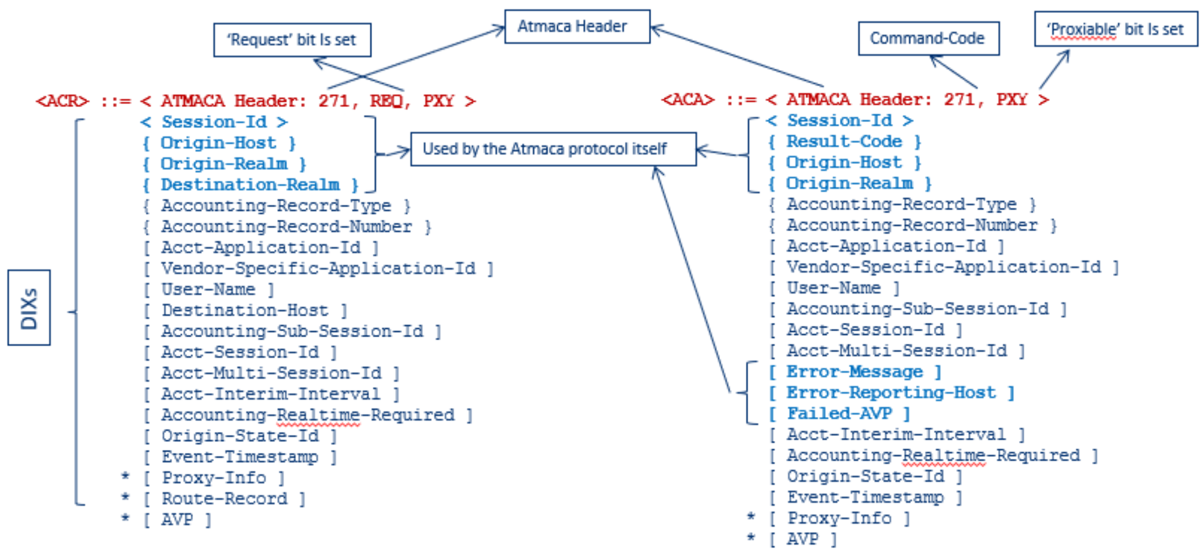
Vendor-ID: Present if the 'V' bit is set in the DIX Flags field. Any vendors or standardisation organisations wishing to implement a vendor-specific ATMACA DIX MUST use their own Vendor-ID along with their privately managed DIX address space, guaranteeing that they will not collide with any other vendor's vendor-specific DIX(es).

4.1.3 ABNF Grammer

Message Naming Conventions:

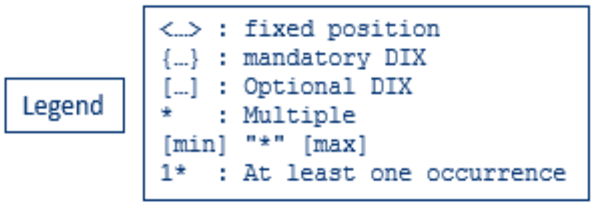
Messages (Figure 4.4) are named with Command-Code names. These are inspired by Diameter command names, which typically include one or more English words followed by the verb "Request" or "Answer". Each English word is delimited by a hyphen. A three-letter acronym for both the request and answer is also normally provided.

For example; The Accounting-Request message is used to transmit the accounting information to the Diameter server, which MUST reply with the Accounting-Answer message to confirm reception. The acronyms for these are ACR and ACA, respectively.



```
<ACR> ::= < ATMACA Header: 271, REQ, PXY >
          < Session-Id >
          { Origin-Host }
          { Origin-Realm }
          { Destination-Realm }
          { Accounting-Record-Type }
          { Accounting-Record-Number }
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ User-Name ]
          [ Destination-Host ]
          [ Accounting-Sub-Session-Id ]
          [ Acct-Session-Id ]
          [ Acct-Multi-Session-Id ]
          [ Acct-Interim-Interval ]
          [ Accounting-Realtime-Required ]
          [ Origin-State-Id ]
          [ Event-Timestamp ]
        * [ Proxy-Info ]
        * [ Route-Record ]
        * [ AVP ]
```

```
<ACA> ::= < ATMACA Header: 271, PXY >
          < Session-Id >
          { Result-Code }
          { Origin-Host }
          { Origin-Realm }
          { Accounting-Record-Type }
          { Accounting-Record-Number }
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ User-Name ]
          [ Accounting-Sub-Session-Id ]
          [ Acct-Session-Id ]
          [ Acct-Multi-Session-Id ]
          [ Error-Message ]
          [ Error-Reporting-Host ]
          [ Failed-AVP ]
          [ Acct-Interim-Interval ]
          [ Accounting-Realtime-Required ]
          [ Origin-State-Id ]
          [ Event-Timestamp ]
        * [ Proxy-Info ]
        * [ AVP ]
```



```
Legend

<…> : fixed position
{…} : mandatory DIX
[…] : Optional DIX
*   : Multiple
[min] "*" [max]
1*  : At least one occurrence
```

Figure 4.4: ATMACA ABNF

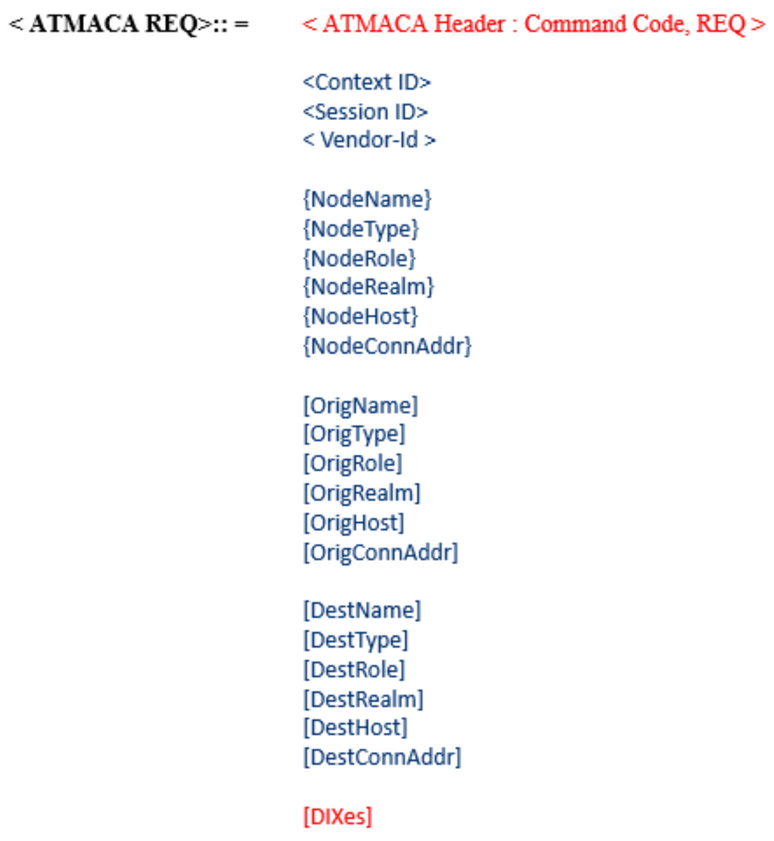
4.1.4 Common DIXs

Figure 4.4 shows common message DIXs, which include:

- Fixed header DIXs: these are basic parameters required for message identification following the header format with a predefined fixed position, such as Command Code or Message Type.
- Mandatory protocol DIXs: the protocol requires additional information to route messages that its obtained from these DIXs. These include the session ID, associating messages to active sessions, and origin and destination realm data.
- Other, optional DIXs: messages aren't required to contain these DIXs, but they are predefined in the dictionary to be used by applications. They contain useful, commonly needed by applications data, such as application ID, username, timestamp or proxy information

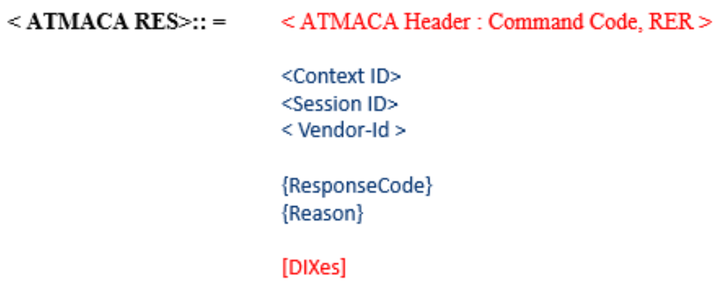
The common DIXs predefined in the base message dictionaries used in each message vary depending on the message type. Figure 4.5 shows the message format for CPDLC requests and Figure 4.6 shows the message format for CPDLC responses. These are meant to showcase the base message formats and their expected common DIXs based on the message type. Response messages require additional Response Code and Reason parameters.

.



```
< ATMACA REQ>:: =     < ATMACA Header : Command Code, REQ >

                      <Context ID>
                      <Session ID>
                      < Vendor-Id >

                      {NodeName}
                      {NodeType}
                      {NodeRole}
                      {NodeRealm}
                      {NodeHost}
                      {NodeConnAddr}

                      [OrigName]
                      [OrigType]
                      [OrigRole]
                      [OrigRealm]
                      [OrigHost]
                      [OrigConnAddr]

                      [DestName]
                      [DestType]
                      [DestRole]
                      [DestRealm]
                      [DestHost]
                      [DestConnAddr]

                      [DIXes]
```

Figure 4.5: ATMACA Request Message Format



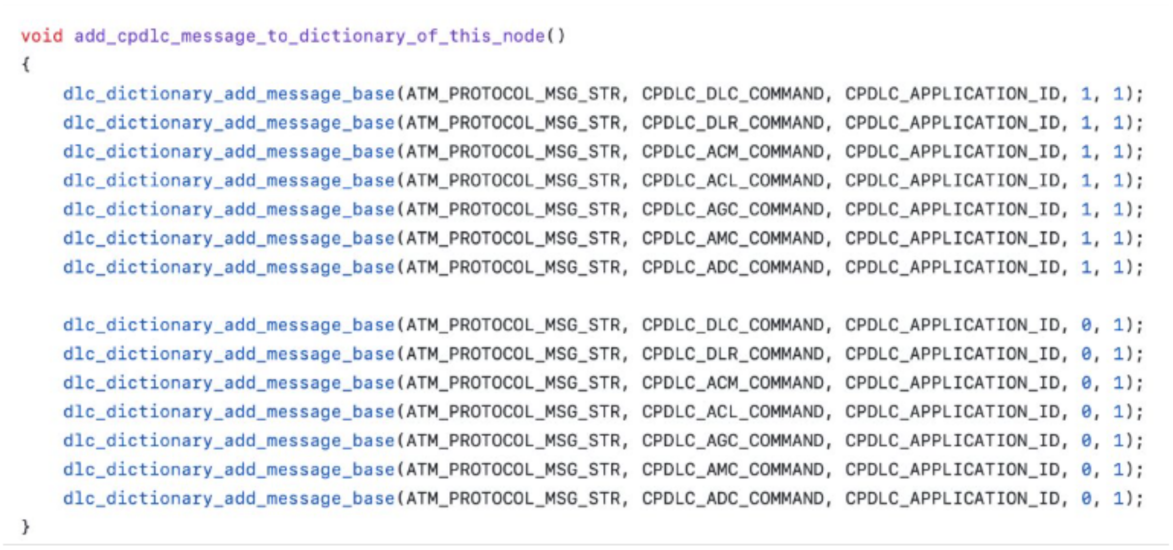
```
< ATMACA RES>:: =     < ATMACA Header : Command Code, RER >

                      <Context ID>
                      <Session ID>
                      < Vendor-Id >

                      {ResponseCode}
                      {Reason}

                      [DIXes]
```

Figure 4.6: ATMACA Response Message Format

4.2 CPDLC Message Definition

The ATMACA stack interface contains a message dictionary to be used for application message registration and later message identification.. All CPDLC messages will have to be registered in the dictionary (

Figure 4.7). This message dictionary includes a CPDLC message list.



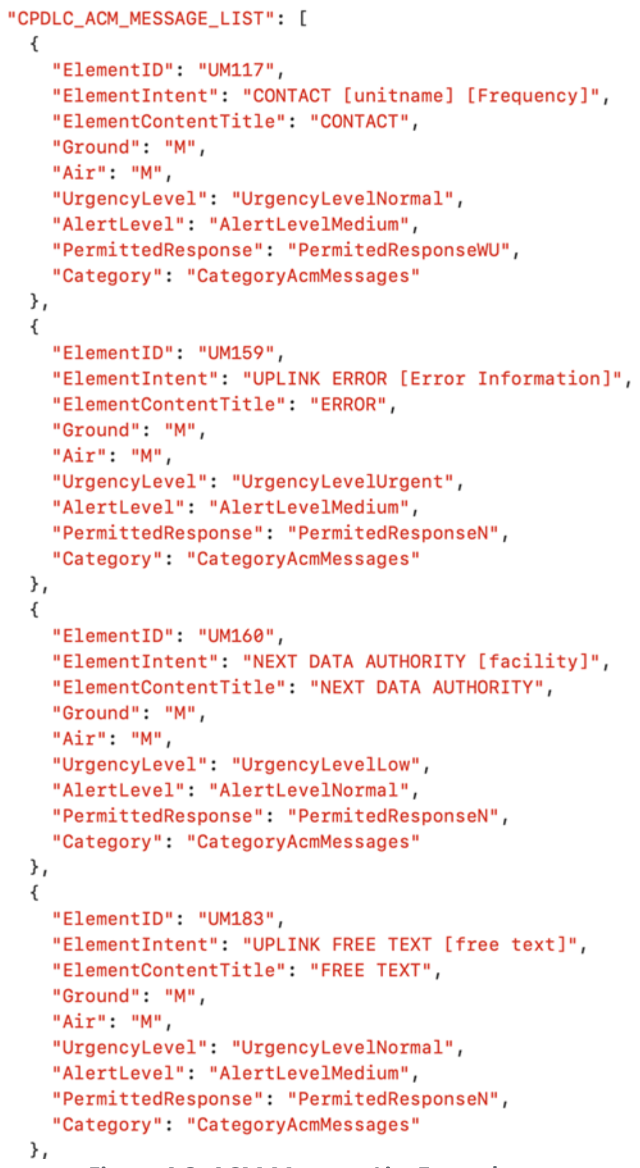
```
void add_cpdlc_message_to_dictionary_of_this_node()
{
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_DLC_COMMAND, CPDLC_APPLICATION_ID, 1, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_DLR_COMMAND, CPDLC_APPLICATION_ID, 1, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_ACM_COMMAND, CPDLC_APPLICATION_ID, 1, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_ACL_COMMAND, CPDLC_APPLICATION_ID, 1, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_AGC_COMMAND, CPDLC_APPLICATION_ID, 1, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_AMC_COMMAND, CPDLC_APPLICATION_ID, 1, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_ADC_COMMAND, CPDLC_APPLICATION_ID, 1, 1);

    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_DLC_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_DLR_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_ACM_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_ACL_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_AGC_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_AMC_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
    dlc_dictionary_add_message_base(ATM_PROTOCOL_MSG_STR, CPDLC_ADC_COMMAND, CPDLC_APPLICATION_ID, 0, 1);
}
```

Figure 4.7: Dictionary Message Addition Example

Example of ACM, AMC, and ACL message lists are shown in Figure 4.8, Figure 4.9, and Figure 4.10 respectively.

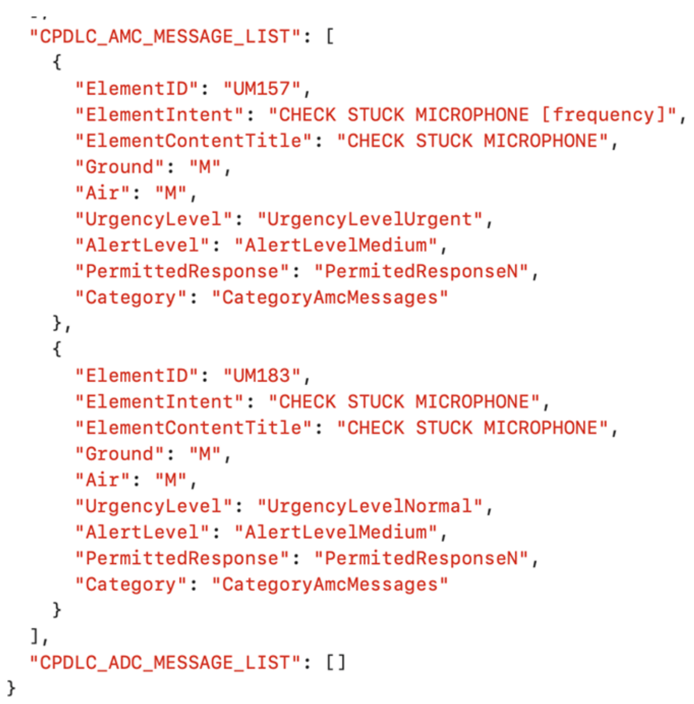
EUROPEAN PARTNERSHIP

Co-funded by the European Union



```
"CPDLC_ACM_MESSAGE_LIST": [
  {
    "ElementID": "UM117",
    "ElementIntent": "CONTACT [unitname] [Frequency]",
    "ElementContentTitle": "CONTACT",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelNormal",
    "AlertLevel": "AlertLevelMedium",
    "PermittedResponse": "PermitedResponseWU",
    "Category": "CategoryAcmMessages"
  },
  {
    "ElementID": "UM159",
    "ElementIntent": "UPLINK ERROR [Error Information]",
    "ElementContentTitle": "ERROR",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelUrgent",
    "AlertLevel": "AlertLevelMedium",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAcmMessages"
  },
  {
    "ElementID": "UM160",
    "ElementIntent": "NEXT DATA AUTHORITY [facility]",
    "ElementContentTitle": "NEXT DATA AUTHORITY",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelLow",
    "AlertLevel": "AlertLevelNormal",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAcmMessages"
  },
  {
    "ElementID": "UM183",
    "ElementIntent": "UPLINK FREE TEXT [free text]",
    "ElementContentTitle": "FREE TEXT",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelNormal",
    "AlertLevel": "AlertLevelMedium",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAcmMessages"
  },
```

Figure 4.8: ACM Message List Example



```
  "CPDLC_AMC_MESSAGE_LIST": [
    {
      "ElementID": "UM157",
      "ElementIntent": "CHECK STUCK MICROPHONE [frequency]",
      "ElementContentTitle": "CHECK STUCK MICROPHONE",
      "Ground": "M",
      "Air": "M",
      "UrgencyLevel": "UrgencyLevelUrgent",
      "AlertLevel": "AlertLevelMedium",
      "PermittedResponse": "PermitedResponseN",
      "Category": "CategoryAmcMessages"
    },
    {
      "ElementID": "UM183",
      "ElementIntent": "CHECK STUCK MICROPHONE",
      "ElementContentTitle": "CHECK STUCK MICROPHONE",
      "Ground": "M",
      "Air": "M",
      "UrgencyLevel": "UrgencyLevelNormal",
      "AlertLevel": "AlertLevelMedium",
      "PermittedResponse": "PermitedResponseN",
      "Category": "CategoryAmcMessages"
    }
  ],
  "CPDLC_ADC_MESSAGE_LIST": []
}
```

Figure 4.9: AMC Message List Example

```
"CPDLC_ACL_MESSAGE_LIST": [
  {
    "ElementID": "UM0",
    "ElementIntent": "UNABLE",
    "ElementContentTitle": "UNABLE",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelNormal",
    "AlertLevel": "AlertLevelMedium",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAclMessages"
  },
  {
    "ElementID": "UM1",
    "ElementIntent": "STANDBY",
    "ElementContentTitle": "STANDBY",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelNormal",
    "AlertLevel": "AlertLevelLow",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAclMessages"
  },
  {
    "ElementID": "UM3",
    "ElementIntent": "ROGER",
    "ElementContentTitle": "ROGER",
    "Ground": "O",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelNormal",
    "AlertLevel": "AlertLevelLow",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAclMessages"
  },
  {
    "ElementID": "UM4",
    "ElementIntent": "AFFIRM",
    "ElementContentTitle": "AFFIRM",
    "Ground": "M",
    "Air": "M",
    "UrgencyLevel": "UrgencyLevelNormal",
    "AlertLevel": "AlertLevelLow",
    "PermittedResponse": "PermitedResponseN",
    "Category": "CategoryAclMessages"
  },
```

Figure 4.10: ACL Message List Example

4.2.1 CPDLC Message Format

CPDLC messages use a standard message format. The elements from this format will be coded as ATMACA DIX to be used by CPDLC application. These messages can contain characters from the IA5 set, including: (0...9), (A...Z), (,), (.), (/), (-), (+), ((), ()) and the space character. These characters are represented using two binary numbers (x/y), from 0 to 7 and from 0 to 15. The message structure contains:

- **Header:** identifies the message
- **Message identification number (MIN):** it is incremented from previous numbers in use and cannot be related to other numbers used by the same aircraft or ATSU.

- **Message Reference Number (MRN):** It equals the MIN in response messages.
- **Timestamp**
- **Indication of logical acknowledgement**
- **Message Elements:** they contain instructions, clearances, requests and relevant information.
- **Message Element Identifier:** the standard message number defined in GOLD.
- **Message data:** operational information to be transmitted.
- **Message element attributes:** additional information
    - **Urgency (URG):** used for queuing (Table 4.1).

Table 4.1: CPDLC Urgency Attribute Types

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

- **Alert (ALRT**) as described in Table 4.2**.**

Table 4.2: Alert Attribute Types

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

Response (RESP) as shown in Table 4.3 and

- Table 4.4**.**

Table 4.3: Response Attribute Types (Uplink)

Type (Uplink response)	Response required	Valid responses	Precedence
W/U	Yes	WILCO, UNABLE, STANDBY permitted, LOGICAL ACKNOWLEDGEMENT (only if required), ERROR (if necessary)	1
A/N	Yes	AFFIRM, NEGATIVE, STANDBY permitted,	2

		LOGICAL ACKNOWLEDGEMENT (only if required), ERROR (if necessary)	
R	Yes	ROGER, UNABLE, STANDBY permitted LOGICAL ACKNOWLEDGEMENT (only if required), ERROR (if necessary)	3
Y	Yes	Any CPDLC downlink message, LOGICAL ACKNOWLEDGEMENT (only if required)	4
N	No, unless logical acknowledgement required	LOGICAL ACKNOWLEDGEMENT (only if required), ERROR (if necessary, only when logical acknowledgement is required)	5

Table 4.4: Response Attribute types (Downlink)

Type (Downlink Response)	Response required	Valid responses	Precedence
Y	Yes	Any CPDLC uplink message, LOGICAL ACKNOWLEDGEMENT (only if required)	1
N	No, unless logical acknowledgement required	LOGICAL ACKNOWLEDGEMENT (only if required), ERROR (if necessary, only when logical acknowledgement is required)	2

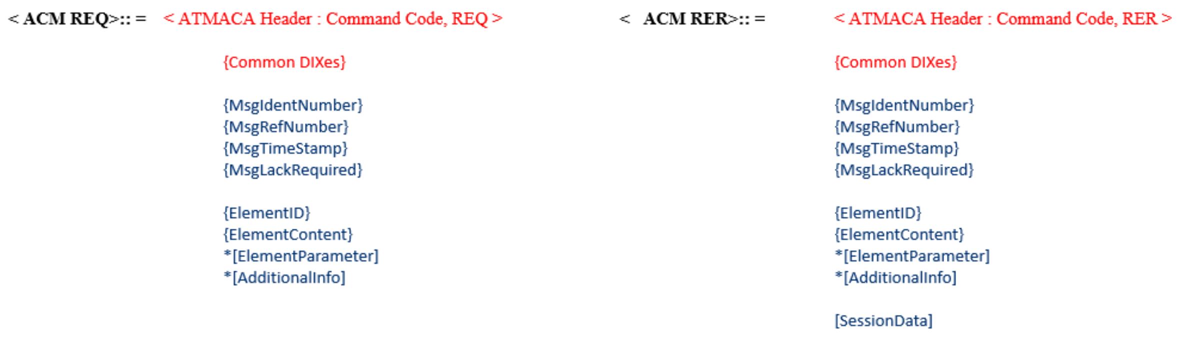
4.2.2 ACM Messages Format

This is an example of how an ACM message (

Figure 4.11) will be defined in ATMACA. The format is similar to other CPDLC messages, as described above.

EUROPEAN PARTNERSHIP

Co-funded by the European Union



```
< ACM REQ>:: =  < ATMACA Header : Command Code, REQ >
                    {Common DIXes}

                    {MsgIdentNumber}
                    {MsgRefNumber}
                    {MsgTimeStamp}
                    {MsgLackRequired}

                    {ElementID}
                    {ElementContent}
                    *[ElementParameter]
                    *[AdditionalInfo]

<  ACM RER>:: =  < ATMACA Header : Command Code, RER >
                    {Common DIXes}

                    {MsgIdentNumber}
                    {MsgRefNumber}
                    {MsgTimeStamp}
                    {MsgLackRequired}

                    {ElementID}
                    {ElementContent}
                    *[ElementParameter]
                    *[AdditionalInfo]

                    [SessionData]
```

Figure 4.11: ACM Message Format

4.2.3 CPDLC Message List

This section defines what messages should be included in the CPDLC application, classified by the service. The required messages are derived by GOLD, Link 2000+, Doc9694 and other applicable standards and official documents (Table 4.5).

Table 4.5: CPDLC Message List

CPDLC Message list	
ACM Uplink Messages	
Message ID	**Format**
UM117	CONTACT [unitname] [frequency]
UM120	MONITOR [unitname] [frequency]
UM159	ERROR [errorInformation]
UM160	NEXT DATA AUTHORITY [facility]
UM183	[free text]
UM227	LOGICAL ACKNOWLEDGEMENT
ACM Downlink Messages	
Message ID	**Format**
DM0	WILCO
DM1	UNABLE
DM2	STANDBY
DM62	ERROR [errorInformation]
DM63	NOT CURRENT DATA AUTHORITY
DM89	MONITORING [unitname] [frequency]
DM98	[freetext] (for additional error information)
DM99	CURRENT DATA AUTHORITY
DM100	LOGICAL ACKNOWLEDGEMENT
DM107	NOT AUTHORISED NEXT DATA AUTHORITY
ACL Uplink Messages	
Message ID	**Format**
UM0	UNABLE
UM1	STANDBY

UM3	ROGER
UM4	AFFIRM
UM5	NEGATIVE
UM19	MAINTAIN [level]
UM20	CLIMB TO [level]
UM23	DESCEND TO [level]
UM74	PROCEED DIRECT TO [position]
UM159	ERROR [errorInformation]
UM162	SERVICE UNAVAILABLE
UM183	[freetext] (for additional error information)
UM190	FLY HEADING [degrees]
UM227	LOGICAL ACKNOWLEDGMENT
ACL Downlink Messages	
Message ID	**Format**
DM0	WILCO
DM1	UNABLE
DM2	STANDBY
DM3	ROGER
DM4	AFFIRM
DM5	NEGATIVE
DM6	REQUEST [level]
DM9	REQUEST CLIMB TO [level]
DM10	REQUEST DESCENT TO [level]
DM22	REQUEST DIRECT TO [position]
DM62	ERROR [errorInformation]
DM65	DUE TO WEATHER
DM66	DUE TO AIRCRAFT PERFORMANCE
DM98	[freetext] (for additional information)
DM100	LOGICAL ACKNOWLEDGMENT
DCL Uplink Messages	
UM73	[departureclearance]
DCL Downlink Messages	
DM25	REQUEST [clearancetype] CLEARANCE
DSC Uplink Messages	
UM183	[freetext] (for information purposes)
DSC Downlink Messages	
DM0	WILCO
DM1	UNABLE
DM2	STANDBY

5 APPLICATION REQUIREMENTS

The main functional requirements related to CPDLC are listed below. They will be taken into account in the CPDLC application design and development, and they will be verified during the CPDLC verification.

5.1 Message Exchange and Communication Handling

- Support for ICAO ATN B2 CPDLC message sets (clearance, instruction, advisory, request, position report).
- Efficient message routing via ATMACA over ATN/IPS, ensuring minimal latency.
- Automatic ATC handover across ATSUs (Air Traffic Service Units) without message loss.
- Error handling mechanisms: Duplicate message detection, message timeouts, and failure recovery.
- Prioritization of safety-critical messages to ensure uninterrupted communication.

5.2 Protocol Adaptation

- Interoperability with legacy ATN/OSI systems and existing aircraft systems via transitional gateways.
- IPv6-compliant ATN/IPS networking for global air traffic communications.
- Use of ATMACA as CPDLC transport layer over ATN/IPS for high-reliability message delivery.
- Support for Quality of Service (QoS) parameters to optimize air-ground message routing.
- Dynamic session management: Establishment, maintenance, and closure of CPDLC connections in a secure manner.

5.3 Safety, Security and Compliance Features

Most security features will be implemented through the base protocol layer

- End-to-end encryption and integrity verification of CPDLC messages via ATN/IPS security layers.
- Authentication mechanisms to prevent spoofing or unauthorized ATC message transmissions.
- Compliance with ICAO Annex 10 Volume III, EUROCAE ED-228A, and EASA regulations for CPDLC over ATN/IPS.
- Real-time monitoring and logging of CPDLC transactions for safety and regulatory auditing.
- Automatic fallback mechanisms: In case of ATMACA or ATN/IPS failures, alternate communication channels (e.g., voice or legacy CPDLC over VDL Mode 2) are available.

5.4 Interoperability and Systems Integration

- Seamless integration with SWIM
- Interoperability with existing CPDLC-capable aircraft operating under FANS 1/A, ATN B1, and ATN B2.
- Dynamic interaction with ATM systems, including Flight Data Processing Systems (FDPS), Air Traffic Flow Management (ATFM) systems, and Controller Working Positions (CWP).
- Scalable architecture supporting both terrestrial and satellite-based CPDLC connectivity.

5.5 Performance and Compliance Requirements

- Latency: End-to-end message delivery within 3-5 seconds under normal operating conditions.
- Availability: 99.99% uptime, ensuring mission-critical communication reliability.
- Data Integrity: No message loss, duplication, or corruption over ATN/IPS.
- Scalability: Capable of handling increased CPDLC traffic as airspace capacity grows.
- Cybersecurity: Implementation of ATN/IPS security protocols (IPsec, TLS, PKI) to prevent cyber threats.
- Interoperability Compliance: Ensuring CPDLC messages over ATMACA are interpretable by both ATN/IPS and legacy ATN/OSI systems during transition phases.

6 USER GUIDE FOR INTEGRATION AND TESTING

6.1 CPDLC Application Testing

As part of the initial development phase, the CPDLC message exchange will be tested prior to its integration with the ATMACA protocol and DLCM. This initial testing will progress alongside the application's development. The objective of this testing is to ensure the proper functioning of the application's components and their logic before proceeding to subsequent testing phases. To achieve this, two peers will be used to check the message interchange implementation based on the protocol. This will help ensure correct operation of the application, including timers, message creation and parsing, message flow, error handling, etc, and other related functions.

6.2 CPDLC Testing within the ATMACA Framework

The CPDLC integration within the framework will be performed. The whole framework will be tested, integrating all its components, from applications to the protocol stack and HMI. This method will ensure that the framework works as intended, showcasing the use cases and measuring KPI achievement. The following user guide provides additional information on how to create, send and receive CPDLC messages. This will be useful for this testing phase.

6.3 CPDLC User Guide

This guide provides users with the necessary information to understand how the CPDLC application works. To this end, it includes details about its various functionalities and instructions on how to use them.

6.3.1 Message Building

The CPDLC application integrates message-building capabilities for all standard services. These builders are coded following the ATMACA protocol message structure and using functions provided by the ATMACA base protocol stack.

The **CpdlcMessageBuilder** provides the core logic for building and parsing CPDLC messages. These messages are essential for node registration, status exchange, and establishing communication contexts between nodes in the ATMACA protocol stack. The module is designed to integrate seamlessly with the rest of the ATMACA CPDLC stack, supporting both message creation for outgoing traffic and parsing for incoming messages.

The application supports a modular approach, with message builders for each CPDLC service type (DLIC, ACM, ACL, AMC, etc.). This allows the CPDLCMessageProcessor to be easily extended for new message types or additional fields. Adding support for a new service involves creating a new builder and data class following the established pattern.

When a user initiates the application to send a request or response message with all necessary DIX data, the message builder is invoked. It serializes its fields into protocol-compliant raw data using DIX elements. Builders handle both mandatory and optional fields, as well as role-specific data for different node types (e.g., mobile client, fixed client, agent).

The CPDLCMessageProcessor implements logic to extract DIX elements from incoming messages and populate the corresponding fields in the message data structures. This ensures that each field is only set once and that lists, and role-specific fields are handled appropriately.

Builders log errors if fields are missing, duplicated, or if there are problems during serialization or parsing. After parsing, each builder can perform compliance checks to ensure the message meets protocol requirements.

Each message data class provides functions to print the contents of messages in a human-readable format, aiding in debugging and development.

The following main functions are implemented:

- **BuildMessageToRaw:** it serialises message data into protocol-compliant raw format. This function builds the message to raw data. The implementation includes common DIXs and CPDLC specific DIXs (MsgIdentNumber, MsgRefNumber, MsgTimeStamp, Msg LackRequired, ElementID, ElementContent, ElementParameter, AdditionalInformation).

- **CheckAndCollectDIXData:** parses DIX elements into structured message data. This method is called per DIX obtained during message parsing and expected to collect message related data from DIX parts of message. It is expected that, this subclass implementation has knowledge about this message structure, content, such that in addition to data collection, it may apply some compliance tests of received message.

- **PrintMessageData:** prints all message fields for debugging and logging.

6.3.2 Application Initialization

The CpdlcServiceHandler module is responsible for the initialisation, lifecycle management, peer and session setup, message dictionary registration, and callback handling of the CPDLC service within ATMACA.

The module encapsulates the core logic for starting, initialising, running, and stopping the CPDLC service. It manages the service's state transitions and message dictionary and provides mechanisms for registering callback functions that process incoming CPDLC requests and responses. This design allows the service to be flexible and decoupled from the specific logic used to handle messages and nodes provided by DLCM, enabling easy integration and testing. It also handles the initialisation (IPL, Init), starting (Start), and stopping (Stop) of the CPDLC service, ensuring correct state transitions and resource management. Furthermore, uses session and context management features from DLCM for CPDLC message exchanges. In addition, it registers all supported CPDLC message types (DLIC, ACM, ACL, etc.) and their makers in the message dictionary, enabling correct parsing and construction of protocol messages.

The service uses an internal status variable to ensure that operations such as initialisation, starting, and stopping are only performed in valid states, preventing invalid transitions and resource leaks.

Its main functions are:

- **IPL:** Initialises the service by setting up the callback interface if the service is in the None or ShuttingDown state. Sets the status to IPLed on success.

- **Init:** Further initializes the service, ensuring that a callback interface is present. Sets the status to Initialised if successful.
- **Start:** Starts the service if it is in the Initialised state. Sets the status to Started.
- **Stop:** Stops or shuts down the service based on the provided reason. Handles state transitions to Stopped or ShuttingDown and cleans up the callback interface as needed.

6.3.3 Message Interchange

Message handling and business logic are implemented in the CpdlcServiceHandler module as well. This module manages message storage, retrieval, and processing for all CPDLC categories. It includes functions for adding, removing, and retrieving messages, as well as processing requests and responses.

The CpdlcServiceHandler module serves as the central component for CPDLC message management and business logic within the ATMACA CPDLC application. It acts as the interface between the protocol/service layer and the application logic, providing a comprehensive set of functions for handling all aspects of CPDLC message processing.

It encompasses the following functionalities:

Message List Management: Maintains categorised lists in JSON for all CPDLC message types (DLIC, ACM, ACL, etc), supporting operations to add, remove, clear, and retrieve message elements. This enables efficient lookup and management of protocol messages based on their type and identifier.

Request and Response Handling: Implements handlers for all standard CPDLC responses (e.g., LACK, WILCO, UNABLE, STANDBY, ERR, AFFIRM, NEGATIVE, ROGER) and provides generic mechanisms for processing both uplink and downlink messages. The module supports both interactive and automated response generation.

Integration with Context and Session Management: Works closely with DLCM to manage node-specific data (e.g., facility, sector, flight information). Contains logic for updating application context based on received messages, such as managing flight lists, sector assignments, and data authority transitions.

Extensibility and Protocol Evolution: The module is structured to allow easy adaptation as the CPDLC protocol and application requirements develop and new functionalities are added.

The following main functions are implemented for their use:

- **Message List Management:**
  - Add, remove, clear, and retrieve CPDLC message elements functions for all supported message types (DLIC, ACM, ACL, etc).
- **Request and Response Handling:**
  - Functions to process incoming CPDLC requests and generate appropriate responses, following the response type.
  - Standard CPDLC response handlers (LACK, WILCO, UNABLE, STANDBY, ERR, AFFIRM, NEGATIVE, ROGER).

- Send request functions, for each message available from the message list.

These functions together enable the module to serve as the core business logic and message management layer for CPDLC operations.

7 Conclusion

This document has outlined the detailed architecture, functionality, and integration design of the Controller–Pilot DataLink Communications (CPDLC) application within the ATMACA framework, developed under the SESAR 3 Joint Undertaking. The CPDLC solution represents a significant advancement in digital air–ground communications, introducing a reliable, standardised, and session-aware communication layer that addresses the growing operational complexity of air traffic management (ATM) in increasingly dense and dynamic airspace environments.

The CPDLC application has been engineered to work seamlessly over an Internet Protocol-based Aeronautical Telecommunication Network (ATN/IPS), enabling ICAO-compliant digital message exchanges between pilots and controllers. Through its integration with the DataLink Context Management (DLCM) middleware, the CPDLC system leverages robust capabilities in session management, role-based access, handover automation, and fault-tolerant communications. These features are essential for supporting uninterrupted controller–pilot data exchanges across airspace boundaries, communication handovers, and diverse network infrastructures.

A central strength of the CPDLC design is its adherence to international standards (e.g., ICAO GOLD), while enhancing flexibility through modularity, extensibility, and dual-mode communication over both TCP and UDP. This hybrid architecture ensures the application can be tailored for varying operational environments, from en-route and terminal areas to complex pre-departure ground procedures. By supporting both strict-pair and hybrid request–response workflows, the solution balances the need for message integrity with the efficiency required for real-time ATM operations.

Throughout the document, the CPDLC architecture has been described, with particular attention to the functional subsystem components and their orchestration through the ATMACA protocol stack. The application supports four key datalink services in its initial implementation: DLIC, ACM, ACL, and DSC, each designed to automate and simplify routine communication procedures while ensuring traceability, compliance, and operational awareness. Furthermore, provisions for enhanced ground services—including D-TAXI, D-STARTUP, and D-PUSHBACK—have been defined, demonstrating a clear path for future capability expansion within the same architectural framework.

The CPDLC system also embodies significant innovation in the handling of session and mobility challenges. By leveraging the mobility-aware capabilities of the ATMACA framework, including its session, context, and connection management modules, the application ensures communication continuity even under complex mobility scenarios such as multi-link operations, network transitions, and ATCo handovers. This makes it particularly suitable for high-density or long-haul operations where traditional voice communication is strained or insufficient.

Moreover, the CPDLC's integration with a user-centric Human–Machine Interface (HMI) allows both controllers and flight crews to manage communications intuitively. Features such as automated logon/logoff procedures, dynamic presence tracking, and session replication across multiple devices or workstations increase the usability, situational awareness, and resilience of the overall system. These benefits directly support SESAR's operational concepts, such as Trajectory-Based Operations (TBO), Green Route Operations (GRO), and flight-centric air traffic control.

This deliverable (D4.2) has provided the technical blueprint necessary for developing, testing, and validating the CPDLC application in the subsequent phases of the ATMACA project, particularly within WP6. The implementation and integration guidelines presented here will facilitate real-world evaluation of the application in both simulated and operational environments, ensuring readiness for future deployment within the European ATM system.

8 REFERENCES

[1] EUROCAE ED-78A Guidelines for Approval of the Provision and Use of Air Traffic Services supported by Data Communications, December 2020

[2] ICAO Doc 9880, Manual on Detailed Technical Specifications for the Aeronautical Telecommunication Network (ATN) using ISO/OSI Standards and Protocols, First Edition - 2010

[3] ICAO Doc 9896, Manual for the ATN using IPS Standards and Protocols, 1st edition

[4] ICAO Doc 9750 (Global Air Navigation Plan for CNS/ATM Systems)

[5] ICAO, *Manual of Air Traffic Services Data Link Applications (Doc 9694)*, 1st ed., Montreal, Canada: International Civil Aviation Organization, 1999.

[6] EUROCONTROL, *ATC Data Link Operational Guidance in Support of DLS Regulation No 29/2009*, Version 6.0, Brussels, Belgium: European Organization for the Safety of Air Navigation, December 2012.

[7] CCITT, *Recommendation T.50 – International Reference Alphabet (IRA) (formerly International Alphabet No. 5 or IA5)*, International Telegraph and Telephone Consultative Committee, Geneva, Switzerland, 1988.

[8] ATMACA-SESAR, Deliverable D3.1, Protocol Design Plan, 2024

[9] ICAO, *Global Operational Data Link Document (GOLD)*, 2nd ed., Montreal, Canada: International Civil Aviation Organization, April 2013

[10] ICAO Doc 10039 (Manual on System Wide Information Management (SWIM) Concept)

[11] ICAO Annex 10 (Aeronautical Telecommunications)

[12] ICAO PANS ATM (Procedures for Air Navigation Services Air Traffic Management)

[13] Easy Access Rules for Air Traffic Management/Air Navigation Services (ATM/ANS) Equipment (Regulations EU 2023/1769 & EU 2023/1768

[14] Eurocontrol, Aeronautical mobile airport communications system datalink (AeroMACS), (2021)

[15] Surveillance Conference (ICNS), Herndon, VA, (2013), pp. 1-13, doi: 10.1109/ICNSurv.2013.6548533

[16] H. Schulzrinne and E. Wedlund, Application-layer mobility using SIP. Mobile Computing and Communications Review, 4(3), pp. 47–57, (2010)

[17] ICAO Annex 11 (Air Traffic Services)

[18] ICAO Doc 4444 (Air Traffic Management)

[19] ATMACA-SESAR, Deliverable D2.1, Review of Current and Future ATM Communication Network, 2024