Controller/Pilot Data Link Communication Application

SARPS Validation Report

February 28, 1997

Prepared by Open Network Solutions

Table of Contents

Controller/Pilot Data Link Communication Application
SARP Validation Report

<u>Overview</u>

The SARPS were validated by the developer using the following criteria:

- **Unambiguous**: Every requirement state should have only one interpretation
- **Completeness:** Inclusion of all significant requirements for functionality, performance, design constraints, attributes, and external interfaces. Definition of the response of software to all classes of input in all classes of situations.
- **Verifiable**: Requirements stated in verifiable or measurable terms.
- **Consistent:** No set of requirements should conflict with another set of requirements.
- **Modifiable**: Can changes be made easily, completely and consistently?
- **Traceable:** Backward traceability depends on each requirement referencing its source in previous documents. Forward traceability depends on each requirement having a unique reference number.
- **Usability During Development**: Requirements stated as to their criticality, modifiability, backward and forward traceability.

Ambiguity

The following section were found to be clear and unambiguous:

- 2.3.3.3 The description of service parameters,
- 2.3.4 Formal definitions of messages,
- 2.3.5.1 Sequence Rules,
- 2.3.5-2 Air State Table,
- 2.3.5-3 Ground State Table

The following sections were found to be generally unambiguous:

- 2.3.5.2 Service Timers. Difficulty found in differentiation between these timers and timers in Chapter 7.
- 2.3.5.4 Air Protocol Description. Specific issues may be found in the section titled Open Issues.
- 2.3.5.5 Ground Protocol Description. Specific issues may be found in the section titled Open Issues.

The following sections were found to be ambiguous:

- 2.3.3 The Abstract Service. Its so abstract as to be useless and only adds confusion as to the functionality of the system.
- 2.3.7 CPDLC User Requirements. No differentiation made between true ‘end’ user requirements and design requirements. Differentiation needed between the automation requirements and ‘end’ user actions. There is also a need for user interface requirements. This section is very confusing.

Completeness

Developer found no system performance requirements in this application’s SARPs. For example, no specification was made for the number of pilots/aircraft the controller’s system must support.

The following sections were found to be complete:

- 2.3.4 Formal definitions of messages,
- 2.3.5.1 Sequence Rules,
- 2.3.5.2 Service Timers.

The following sections were found to be generally complete:

- 2.3.5-2 Air State Table,
- 2.3.5-3 Ground State Table,
- 2.3.5.4 Air Protocol Description,
- 2.3.5.5 Ground Protocol Description.

Specific issues may be found in the section titled Open Issues.

The following sections were found to be incomplete:

- 2.3.6 Communication Requirements. Definition of fields needed by the communication service is missing. (i.e. mode, receive type, ae qualifiers, esi, etc.) These need to be specified for each service primitive.
- 2.3.7 CPDLC User Requirements. Definition is needed in the exact classes of response the automation is to make to a given input.

Verifiable

The following sections were found to be verifiable:

- 2.3.5-2 Air State Table,
- 2.3.5-3 Ground State Table,
- 2.3.5.4 Air Protocol Description,
- 2.3.5.5 Ground Protocol Description.
- 2.3.4 Formal definitions of messages,
- 2.3.5.1 Sequence Rules,
- 2.3.5.2 Service Timers.

The following section was found to be unverifiable:

- 2.3.7 CPDLC User Requirements. Definition is needed in the exact classes of response the automation is to make to a given input. Verifying the software product meets the automation requirement is not possible.

Consistency

The SARPS generally exhibit consistency throughout the document.

Traceability

Generally the application SARP needs to add backward traceability. The origin of the requirement needs to be specified. For example, the allocation of response time for the application timer results from the overall communication response time of ?.

Usability During Development

The document desperately needs a detailed table of contents to assist in finding all details of specific requirement. For example, the developer must find all processing details for a CPDLC Start Request by referencing sections 2.3.3.3, 2.3.5, 2.3.5.3.7, 2.3.5.4, 2.3.5.5.7, 2.3.5.7, 2.3.7.3, 2.3.7.4.1, 2.3.7.5.1. There is no cross reference to assist the developer in finding the references.

The document needs a glossary to define terms. For example, definitions are needed for PDUs, ASN.1 message structure, etc.

The document needs a revision log with dates so developers can verify the copy of SARPS they are developing.

Open Issues

Following are questions raised by the developer not addressed by the SARPs.

- Is there a condition where a pilot will not have a data authority?
- How many pilots/aircraft does a controller need to tract through the system?
- Can more than one pilot per aircraft be comminicating with the controller?
- Can the pilot have multiple dialogues at one time?
- Can a controller have mulitple functions at a time? (ie ground forward, CPDLC dialogues, DSC dialogues)
- What data is locally edited?
- How much data can be entered by the user pre-dialogue?
- How do the service timers and the chapter 7 timers interact?
- Are ground system users always controllers?
- Are CPDLC functions mutually exclusive?
- How many ground/ground dialogues can a controller start?
- Does system need to maintain current data authority association after provider abort?
- Does the system need to queue dialogues? Or is queing needed for message elements?
- What specific actions should be taken if conditions are not equal to state table? For example, in the Ground state table. If a D_END indication is received and DSC=FALSE there is no action described in the SARPS. If you interpret it as a ‘cannot occur’ condition it would mean aborting the dialogue. Another example, after the dialogue is correctly established for CPDLC Air/Ground there is no specific action specified for APDU not equal to Uplink message. If you interpret it as a ‘cannot occur’ it would mean aborting the dialogue.
- Some of the language in chapter 5 needs to be more specific and consistent. For example. 2.3.5.4.4.1 in paragraphs b and d, the actions described look contridictory as to the location of the not-permitted-PDU reason.

Test Cases for State Tables

Air State Tables

Event	**State**	**Results**
CPDLC_start_req	Idle	Verified receipt of event and state set correctly.

Test Case: Air Start with Ground Server

Event Sequence Test (Figure 2.3.5-1)

Origin	**Event**	**SARP**
Air	CPDLC_start_request	2.3.5.3.7
DS	D_start_req	
DS	D_start_ind	2.3.5.5.2.1
Ground	CPDLC_start_ind	CH 5=? 2.3.7.5.1.2
Ground	CPDLC_start_rsp	2.3.5.5.8.1 2.3.7.5.1.2
DS	D_start_rsp	
DS	D_start_cnf	2.3.5.5.3.1
AIR	CPDLC_start_cnf	CH 5=? 2.3.7.4.1.3

Test Case: Air Message Start with Ground Server

Event Sequence Test (Figure 2.3.5-4)

Origin	**Event**	**SARP**
Air	CPDLC_msg_req	2.3.5.3.10.1
DS	D_start_req	
DS	D_start_ind	2.3.5.5.4.1
Ground	CPDLC_msg_ind	CH 5=?

Test Case: Ground Message Start with Ground Server

Event Sequence Test (Figure 2.3.5-5)

Origin	**Event**	**SARP**
Ground	CPDLC_msg_req	2.3.5.3.10.1
DS	D_start_req	
DS	D_start_ind	2.3.5.3.4.1
Air	CPDLC_msg_ind	?

Test Case: Ground End Service with Ground Server

Event Sequence Test (Figure 2.3.5-6)

Origin	**Event**	**SARP**
Ground	CPDLC_end_req	2.3.5.3.7
DS	D_end_req	
DS	D_end_ind	2.3.5.5.2.1
Air	CPDLC_end_ind	CH 5=?
Air	CPDLC_end_rsp	2.3.5.3.11.1
DS	D_end_rsp	
DS	D_end_cnf	2.3.5.5.6.1
AIR	CPDLC_end_cnf	CH 5=?

Test Case: Ground Start with Air Server

Event Sequence Test (Figure 2.3.5-2)

Origin	**Event**	**SARP**
Ground	CPDLC_start_request	2.3.5.5.7
DS	D_start_req	
DS	D_start_ind	2.3.5.5.8.1
Air	CPDLC_start_ind	CH 5=?
Air	CPDLC_start_rsp	2.3.5.5.8.1
DS	D_start_rsp	
DS	D_start_cnf	2.3.5.5.3.1
Ground	CPDLC_start_cnf	CH 5=?

Test Case: Air Message Start with Air Server

Event Sequence Test (Figure 2.3.5-4)

Origin	**Event**	**SARP**
Air	CPDLC_msg_req	2.3.5.3.10.1
DS	D_start_req	
DS	D_start_ind	2.3.5.5.4.1
Ground	CPDLC_msg_ind	CH 5=?

Test Case: Ground Message Start with Air Server

Event Sequence Test (Figure 2.3.5-5)

Origin	**Event**	**SARP**
Ground	CPDLC_msg_req	2.3.5.3.10.1

DS	D_start_req	
DS	D_start_ind	2.3.5.3.4.1
Air	CPDLC_msg_ind	?

Test Case: Ground End Service with Air Server

Event Sequence Test (Figure 2.3.5-6)

Origin	**Event**	**SARP**
Ground	CPDLC_end_req	2.3.5.3.7
DS	D_end_req	
DS	D_end_ind	2.3.5.5.2.1
Air	CPDLC_end_ind	CH 5=?
Air	CPDLC_end_rsp	2.3.5.3.11.1
DS	D_end_rsp	
DS	D_end_cnf	2.3.5.5.6.1
AIR	CPDLC_end_cnf	CH 5=?

Test Case: Forward Service - Versions Equal

Event Sequence Test (Figure 2.3.5-8)

Origin	**Event**	**SARP**
Ground	CPDLC_forward_request	2.3.5.5.13
DS	D_start_req	
DS	D_start_ind	2.3.5.5.8.1?

Ground	CPDLC_forward_ind	CH 5=?
DS	D_start_rsp	
DS	D_start_cnf	2.3.5.5.3.1?
Ground	CPDLC_forward_cnf	CH 5=?

<u>**Test Case**: Forward Service - Versions Not Equal</u>

Event Sequence Test (Figure 2.3.5-9)

Origin	**Event**	**SARP**
Ground	CPDLC_forward_request	2.3.5.5.13
DS	D_start_req	
DS	D_start_ind	2.3.5.5.8.1?
DS	D_start_rsp	
DS	D_start_cnf	2.3.5.5.3.1?
Ground	CPDLC_forward_cnf	CH 5=?

<u>**Test Case**: CPDLC Provider Abort - Air ASE Abort</u>

Event Sequence Test (Figure 2.3.5-13)

Origin	**Event**	**SARP**
Air	CPDLC_P_abort_ind	CH 5=?
DS	D_ABORT_req	CH 5=?
DS	D_ABORT_ind	2.3.5.5.15

Ground	CPDLC_P_abort_ind	CH 5=?

Test Case: CPDLC Provider Abort - Ground User Abort

Event Sequence Test (Figure 2.3.5-11)

Origin	**Event**	**SARP**
Ground	CPDLC_user_abort_req	2.3.5.5.14
DS	D_ABORT_req	
DS	D_ABORT_ind	2.3.5.3.14
Air	CPDLC_user_abort_ind	

Test Case: CPDLC Provider Abort - Air User Abort

Event Sequence Test (Figure 2.3.5-10)

Origin	**Event**	**SARP**
Air	CPDLC_user_abort_req	2.3.5.3.13
DS	D_ABORT_req	
DS	D_ABORT_ind	2.3.5.5.15
Ground	CPDLC_user_abort_ind	