

Industrial communication networks — Fieldbus specifications —

Part 4-2: Data-link layer protocol specification — Type 2 elements

ICS 25.040.40; 35.100.20

National foreword

This British Standard is the UK implementation of EN 61158-4-2:2008. It is identical with IEC 61158-4-2:2007. Together with all of the other sections of BS EN 61158-4, it supersedes BS EN 61158-4:2004 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications — Process measurement and control, including Fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 May 2008

© BSI 2008

ISBN 978 0 580 61575 7

Amendments/corrigenda issued since publication

Date	Comments

English version

**Industrial communication networks -
Fieldbus specifications -
Part 4-2: Data-link layer protocol specification -
Type 2 elements
(IEC 61158-4-2:2007)**

Réseaux de communication industriels -
Spécifications des bus de terrain -
Partie 4-2: Spécification des protocoles
des couches de liaison de données -
Éléments de type 2
(CEI 61158-4-2:2007)

Industrielle Kommunikationsnetze -
Feldbusse -
Teil 4-2: Protokollspezifikation des
Data Link Layer (Sicherheitsschicht) -
Typ 2-Elemente
(IEC 61158-4-2:2007)

This European Standard was approved by CENELEC on 2008-02-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 65C/474/FDIS, future edition 1 of IEC 61158-4-2, prepared by SC 65C, Industrial networks, of IEC TC 65, Industrial-process measurement, control and automation, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61158-4-2 on 2008-02-01.

This and the other parts of the EN 61158-4 series supersede EN 61158-4:2004.

With respect to EN 61158-4:2004 the following changes were made:

- deletion of Type 6 fieldbus, and the placeholder for a Type 5 fieldbus data-link layer, for lack of market relevance;
- addition of new fieldbus types;
- partition into multiple parts numbered 4-1, 4-2, ..., 4-19.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2008-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2011-02-01

NOTE Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in type combinations as specified explicitly in the EN 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

IEC and CENELEC draw attention to the fact that it is claimed that compliance with this standard may involve the use of patents as follows, where the [xx] notation indicates the holder of the patent right:

Type 2 and possibly other types:

US 5,400,331	[RA]	Communication network interface with screeners for incoming messages
US 5,471,461	[RA]	Digital communication network with a moderator station election process
US 5,491,531	[RA]	Media access controller with a shared class message delivery capability
US 5,493,571	[RA]	Apparatus and method for digital communications with improved delimiter detection
US 5,537,549	[RA]	Communication network with time coordinated station activity by time slot and periodic interval number
US 5,553,095	[RA]	Method and apparatus for exchanging different classes of data during different time intervals

IEC and CENELEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holders of these patent rights are registered with IEC. Information may be obtained from:

[RA]: Rockwell Automation, Inc.
1201 S. Second Street
Milwaukee, WI 53204
USA
Attention: Intellectual Property Dept.

Attention is drawn to the possibility that some of the elements of this standard may be the subject of patent rights other than those identified above. IEC and CENELEC shall not be held responsible for identifying any or all such patent rights.

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61158-4-2:2007 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION	8
1 Scope	9
1.1 General	9
1.2 Specifications	9
1.3 Procedures	9
1.4 Applicability	10
1.5 Conformance	10
2 Normative references	10
3 Terms, definitions, symbols and abbreviations	11
3.1 Reference model terms and definitions	11
3.2 Service convention terms and definitions	13
3.3 Common terms and definitions	14
3.4 Additional Type 2 definitions	15
3.5 Type 2 symbols and abbreviations	22
4 Overview of the DL-protocol	23
4.1 General	23
4.2 Services provided by the DL	25
4.3 Structure and definition of DL-addresses	26
4.4 Services assumed from the PhL	28
4.5 Functional classes	30
5 General structure and encoding of PhIDUs and DLPDUs and related elements of procedure	31
5.1 Overview	31
5.2 Media access procedure	31
5.3 DLPDU structure and encoding	34
5.4 Lpacket components	38
5.5 DLPDU procedures	40
5.6 Summary of DLL support services and objects	41
6 Specific DLPDU structure, encoding and procedures	43
6.1 Modeling language	43
6.2 DLS user services	45
6.3 Generic tag Lpacket	51
6.4 Moderator Lpacket	52
6.5 Time distribution Lpacket	53
6.6 UCMM Lpacket	56
6.7 Keeper UCMM Lpacket	56
6.8 TUI Lpacket	57
6.9 Link parameters Lpacket and tMinus Lpacket	58
6.10 I'm-alive Lpacket	59
6.11 Ping Lpackets	61
6.12 WAMI Lpacket	63
6.13 Debug Lpacket	63
6.14 IP Lpacket	64
6.15 Ethernet Lpacket	64
7 Objects for station management	64
7.1 General	64

7.2	ControlNet object	65
7.3	Keeper object	75
7.4	Scheduling object	97
7.5	TCP/IP Interface object	108
7.6	Ethernet link object	117
7.7	DeviceNet object	123
7.8	Connection configuration object (CCO)	131
8	Other DLE elements of procedure	151
8.1	Network attachment monitor (NAM)	151
8.2	Calculating link parameters	158
9	Detailed specification of DL components	166
9.1	General	166
9.2	Access control machine (ACM)	166
9.3	TxLLC	183
9.4	RxLLC	187
9.5	Transmit machine (TxM)	190
9.6	Receive machine (RxM)	193
9.7	Serializer	199
9.8	Deserializer	200
9.9	DLL management	201
Annex A	(normative) Indicators and switches	204
A.1	Purpose	204
A.2	Indicators	204
A.3	Switches	215
Annex ZA	(normative) Normative references to international publications with their corresponding European publications	219
Bibliography		217
INDEX		218
Figure 1	– Relationships of DLSAPs, DLSAP-addresses and group DL-addresses	14
Figure 2	– Data-link layer internal architecture	24
Figure 3	– Basic structure of a MAC ID address	26
Figure 4	– Basic structure of a generic tag address	26
Figure 5	– Basic structure of a fixed tag address	27
Figure 6	– M_symbols and Manchester encoding at 5 MHz (informative)	28
Figure 7	– NUT structure	32
Figure 8	– Media access during scheduled time	32
Figure 9	– Media access during unscheduled time	33
Figure 10	– DLPDU format	34
Figure 11	– Aborting a DLPDU during transmission	38
Figure 12	– Lpacket format	38
Figure 13	– Generic tag Lpacket format	39
Figure 14	– Fixed tag Lpacket format	40
Figure 15	– Goodness parameter of TimeDist_Lpacket	54
Figure 16	– Example I'm alive processing algorithm	

Figure 17 – Keeper CRC algorithm	81
Figure 18 – Keeper object power-up state diagram	93
Figure 19 – Keeper object operating state diagram	94
Figure 20 – Synchronized network change processing	97
Figure 21 – State transition diagram for TCP/IP Interface object	117
Figure 22 – Connection configuration object edit flowchart.....	151
Figure 23 – NAM state machine	152
Figure A.1 – Non redundant network status indicator labeling	208
Figure A.2 – Redundant network status indicator labeling	209
Table 1 – Data-link layer components	23
Table 2 – MAC ID addresses allocation	26
Table 3 – Fixed tag service definitions	27
Table 4 – Data encoding rules	28
Table 5 – M Data symbols	29
Table 6 – Truth table for ph_status_indication.....	30
Table 7 – FCS length, polynomials and constants	36
Table 8 – DLL support services and objects	42
Table 9 – Elementary data types.....	45
Table 10 – DLL events	50
Table 11 – Time distribution priority	55
Table 12 – Format of the TUI Lpacket.....	57
Table 13 – ControlNet object class attributes	66
Table 14 – ControlNet object instance attributes	66
Table 15 – TUI status flag bits	70
Table 16 – Channel state bits	71
Table 17 – ControlNet object common services.....	73
Table 18 – ControlNet object class specific services	74
Table 19 – Keeper object revision history	76
Table 20 – Keeper object class attributes	76
Table 21 – Keeper object instance attributes	76
Table 22 – Keeper operating state definitions	79
Table 23 – Port status flag bit definitions	79
Table 24 – TUI status flag bits	80
Table 25 – Keeper attributes.....	83
Table 26 – Memory requirements (in octets) for the Keeper attributes.....	83
Table 27 – Keeper object common services	84
Table 28 – Keeper object class specific services	85
Table 29 – Service error codes	86
Table 30 – Wire order format of the TUI Lpacket.....	90
Table 31 – Service error codes	91
Table 32 – Keeper object operating states	91
Table 33 – Keeper object state event matrix	95

Table 34 – Scheduling object class attributes	98
Table 35 – Scheduling object instance attributes	99
Table 36 – Scheduling object common services	99
Table 37 – Status error descriptions for Create	100
Table 38 – Status error descriptions for Delete and Kick_Timer	101
Table 39 – Scheduling object class specific services	101
Table 40 – Status error descriptions for Read	103
Table 41 – Status error descriptions for Conditional_Write	104
Table 42 – Status error descriptions for Forced_Write	104
Table 43 – Status error descriptions for Change_Start	105
Table 44 – Status error descriptions for Break_Connections	105
Table 45 – Status error descriptions for Change_Complete	106
Table 46 – Status error descriptions for Restart_Connections	107
Table 47 – TCP/IP Interface object class attributes	108
Table 48 – TCP/IP Interface object instance attributes	109
Table 49 – Status bits	111
Table 50 – Configuration capability bits	111
Table 51 – Configuration control bits	112
Table 52 – Example path	112
Table 53 – Interface configuration components	113
Table 54 – Alloc control values	114
Table 55 – TCP/IP Interface object common services	115
Table 56 – Get_Attribute_All reply format	116
Table 57 – Ethernet link object revision history	117
Table 58 – Ethernet link object class attributes	118
Table 59 – Ethernet link object instance attributes	118
Table 60 – Interface flags bits	120
Table 61 – Control bits	122
Table 62 – Ethernet Link object common services	122
Table 63 – Ethernet Link object class specific services	123
Table 64 – DeviceNet object revision history	123
Table 65 – DeviceNet object class attributes	124
Table 66 – DeviceNet object instance attributes	124
Table 67 – Bit rate attribute values	126
Table 68 – BOI attribute values	127
Table 69 – Diagnostic counters bit description	128
Table 70 – DeviceNet object common services	129
Table 71 – Reset service parameter	129
Table 72 – Reset service parameter values	130
Table 73 – DeviceNet object class specific services	130
Table 74 – Connection configuration object revision history	131
Table 75 – Connection configuration object class attributes	131
Table 76 – Format number values	133

Table 77 – Connection configuration object instance attributes	134
Table 78 – Originator connection status values	136
Table 79 – Target connection status values	137
Table 80 – Connection flags	137
Table 81 – I/O mapping formats	139
Table 82 – Connection configuration object common services	140
Table 83 – Get_Attribute_All error codes	140
Table 84 – Get_Attribute_All response	141
Table 85 – Set_Attribute_All error codes	142
Table 86 – Set_Attribute_All request	143
Table 87 – Create request parameters	144
Table 88 – Create error codes	145
Table 89 – Delete error codes	145
Table 90 – Restore error codes	145
Table 91 – Connection configuration object class specific services	146
Table 92 – Kick_Timer error codes	146
Table 93 – Open_Connection error codes	147
Table 94 – Close_Connection error codes	147
Table 95 – Stop_Connection error codes	147
Table 96 – Change_Start error codes	148
Table 97 – Get_Status service parameter	148
Table 98 – Get_Status service response	148
Table 99 – Get_Status service error codes	149
Table 100 – Change_Complete service parameter	149
Table 101 – Change_Complete service error codes	149
Table 102 – Audit_Changes service parameter	150
Table 103 – Audit_Changes service error codes	150
Table 104 – NAM states	152
Table 105 – Default link parameters	153
Table 106 – PhL timing characteristics	159
Table A.1 – Module status indicator	205
Table A.2 – Network status indicators	206
Table A.3 – Network status indicator	210
Table A.4 – Network status indicator	212
Table A.5 – Combined module/network status indicator	213
Table A.6 – I/O status indicator	214
Table A.7 – Bit rate switch encoding	216

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the “three-layer” fieldbus reference model described in IEC/TR 61158-1.

The data-link protocol provides the data-link service by making use of the services available from the physical layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer data-link entities (DLEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- a) as a guide for implementors and designers;
- b) for use in the testing and procurement of equipment;
- c) as part of an agreement for the admittance of systems into the open systems environment;
- d) as a refinement to the understanding of time-critical communications within OSI.

This standard is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this standard together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-2: Data-link layer protocol specification – Type 2 elements

1 Scope

1.1 General

The data-link layer provides basic time-critical messaging communications between devices in an automation environment.

This protocol provides communication opportunities to all participating data-link entities, sequentially and in a cyclic synchronous manner. Foreground scheduled access is available for time-critical activities together with background unscheduled access for less critical activities.

Deterministic and synchronized transfers can be provided at cyclic intervals up to 1 ms and device separations of 25 km. This performance is adjustable dynamically and on-line by re-configuring the parameters of the local link whilst normal operation continues. By similar means, DL connections and new devices may be added or removed during normal operation.

This protocol provides means to maintain clock synchronization across an extended link with a precision better than 10 μ s.

This protocol optimizes each access opportunity by concatenating multiple DLSDUs and associated DLPCI into a single DLPDU, thereby improving data transfer efficiency for data-link entities that actively source multiple streams of data.

The maximum system size is an unlimited number of links of 99 nodes, each with 255 DLSAP-addresses. Each link has a maximum of 2^{24} related peer and publisher DLCEPs.

1.2 Specifications

This standard specifies

- a) procedures for the timely transfer of data and control information from one data-link user entity to a peer user entity, and among the data-link entities forming the distributed data-link service provider;
- b) the structure of the fieldbus DLPDUs used for the transfer of data and control information by the protocol of this standard, and their representation as physical interface data units.

1.3 Procedures

The procedures are defined in terms of

- a) the interactions between peer DL-entities (DLEs) through the exchange of fieldbus DLPDUs;
- b) the interactions between a DL-service (DLS) provider and a DLS-user in the same system through the exchange of DLS primitives;
- c) the interactions between a DLS-provider and a Ph-service provider in the same system through the exchange of Ph-service primitives.

1.4 Applicability

These procedures are applicable to instances of communication between systems which support time-critical communications services within the data-link layer of the OSI or fieldbus reference models, and which require the ability to interconnect in an open systems interconnection environment.

Profiles provide a simple multi-attribute means of summarizing an implementation's capabilities, and thus its applicability to various time-critical communications needs.

1.5 Conformance

This standard also specifies conformance requirements for systems implementing these procedures. This standard does not contain tests to demonstrate compliance with such requirements.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

IEC 61158-2 (Ed.4.0), *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-3-2, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*

IEC 61158-5-2, *Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements*

IEC 61158-6-2, *Industrial communication networks – Fieldbus specifications – Part 6-2: Application layer protocol specification – Type 2 elements*

IEC 61784-3-2, *Industrial communication networks – Profiles – Part 3-2: Functional safety fieldbuses – Additional specifications for CPF 2*

IEC 62026-3¹, *Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet*

ISO/IEC 3309, *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures – Frame structure*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8802-3, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

¹ To be published.

ISO/IEC 10731, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ISO 11898:1993², *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, the following terms, definitions, symbols and abbreviations apply.

3.1 Reference model terms and definitions

This standard is based in part on the concepts developed in ISO/IEC 7498-1 and ISO/IEC 7498-3, and makes use of the following terms defined therein.

3.1.1	called-DL-address	[ISO/IEC 7498-3]
3.1.2	calling-DL-address	[ISO/IEC 7498-3]
3.1.3	centralized multi-end-point-connection	[ISO/IEC 7498-1]
3.1.4	correspondent (N)-entities correspondent DL-entities (N=2) correspondent Ph-entities (N=1)	[ISO/IEC 7498-1]
3.1.5	demultiplexing	[ISO/IEC 7498-1]
3.1.6	DL-address	[ISO/IEC 7498-3]
3.1.7	DL-address-mapping	[ISO/IEC 7498-1]
3.1.8	DL-connection	[ISO/IEC 7498-1]
3.1.9	DL-connection-end-point	[ISO/IEC 7498-1]
3.1.10	DL-connection-end-point-identifier	[ISO/IEC 7498-1]
3.1.11	DL-connection-mode transmission	[ISO/IEC 7498-1]
3.1.12	DL-connectionless-mode transmission	[ISO/IEC 7498-1]
3.1.13	DL-data-sink	[ISO/IEC 7498-1]
3.1.14	DL-data-source	[ISO/IEC 7498-1]
3.1.15	DL-duplex-transmission	[ISO/IEC 7498-1]
3.1.16	DL-facility	[ISO/IEC 7498-1]
3.1.17	DL-local-view	[ISO/IEC 7498-3]
3.1.18	DL-name	[ISO/IEC 7498-3]
3.1.19	DL-protocol	[ISO/IEC 7498-1]
3.1.20	DL-protocol-connection-identifier	[ISO/IEC 7498-1]
3.1.21	DL-protocol-control-information	[ISO/IEC 7498-1]

² A newer edition of this standard has been published, but only the cited edition applies.

3.1.22	DL-protocol-data-unit	[ISO/IEC 7498-1]
3.1.23	DL-protocol-version-identifier	[ISO/IEC 7498-1]
3.1.24	DL-relay	[ISO/IEC 7498-1]
3.1.25	DL-service-connection-identifier	[ISO/IEC 7498-1]
3.1.26	DL-service-data-unit	[ISO/IEC 7498-1]
3.1.27	DL-simplex-transmission	[ISO/IEC 7498-1]
3.1.28	DL-subsystem	[ISO/IEC 7498-1]
3.1.29	DL-user-data	[ISO/IEC 7498-1]
3.1.30	flow control	[ISO/IEC 7498-1]
3.1.31	layer-management	[ISO/IEC 7498-1]
3.1.32	multiplexing	[ISO/IEC 7498-3]
3.1.33	naming-(addressing)-authority	[ISO/IEC 7498-3]
3.1.34	naming-(addressing)-domain	[ISO/IEC 7498-3]
3.1.35	naming-(addressing)-subdomain	[ISO/IEC 7498-3]
3.1.36	(N)-entity DL-entity Ph-entity	[ISO/IEC 7498-1]
3.1.37	(N)-interface-data-unit DL-service-data-unit (N=2) Ph-interface-data-unit (N=1)	[ISO/IEC 7498-1]
3.1.38	(N)-layer DL-layer (N=2) Ph-layer (N=1)	[ISO/IEC 7498-1]
3.1.39	(N)-service DL-service (N=2) Ph-service (N=1)	[ISO/IEC 7498-1]
3.1.40	(N)-service-access-point DL-service-access-point (N=2) Ph-service-access-point (N=1)	[ISO/IEC 7498-1]
3.1.41	(N)-service-access-point-address DL-service-access-point-address (N=2) Ph-service-access-point-address (N=1)	[ISO/IEC 7498-1]
3.1.42	peer-entities	[ISO/IEC 7498-1]
3.1.43	Ph-interface-control-information	[ISO/IEC 7498-1]
3.1.44	Ph-interface-data	[ISO/IEC 7498-1]
3.1.45	primitive name	[ISO/IEC 7498-3]
3.1.46	reassembling	[ISO/IEC 7498-1]
3.1.47	recombining	[ISO/IEC 7498-1]

3.1.48	reset	[ISO/IEC 7498-1]
3.1.49	responding-DL-address	[ISO/IEC 7498-3]
3.1.50	routing	[ISO/IEC 7498-1]
3.1.51	segmenting	[ISO/IEC 7498-1]
3.1.52	sequencing	[ISO/IEC 7498-1]
3.1.53	splitting	[ISO/IEC 7498-1]
3.1.54	synonymous name	[ISO/IEC 7498-3]
3.1.55	systems-management	[ISO/IEC 7498-1]

3.2 Service convention terms and definitions

This standard also makes use of the following terms defined in ISO/IEC 10731 as they apply to the data-link layer:

3.2.1	acceptor
3.2.2	asymmetrical service
3.2.3	confirm (primitive); requestor.deliver (primitive)
3.2.4	deliver (primitive)
3.2.5	DL-confirmed-facility
3.2.6	DL-facility
3.2.7	DL-local-view
3.2.8	DL-mandatory-facility
3.2.9	DL-non-confirmed-facility
3.2.10	DL-provider-initiated-facility
3.2.11	DL-provider-optional-facility
3.2.12	DL-service-primitive; primitive
3.2.13	DL-service-provider
3.2.14	DL-service-user
3.2.15	DL-user-optional-facility
3.2.16	indication (primitive) acceptor.deliver (primitive)
3.2.17	multi-peer
3.2.18	request (primitive); requestor.submit (primitive)
3.2.19	requestor
3.2.20	response (primitive); acceptor.submit (primitive)

3.2.21 submit (primitive)

3.2.22 symmetrical service

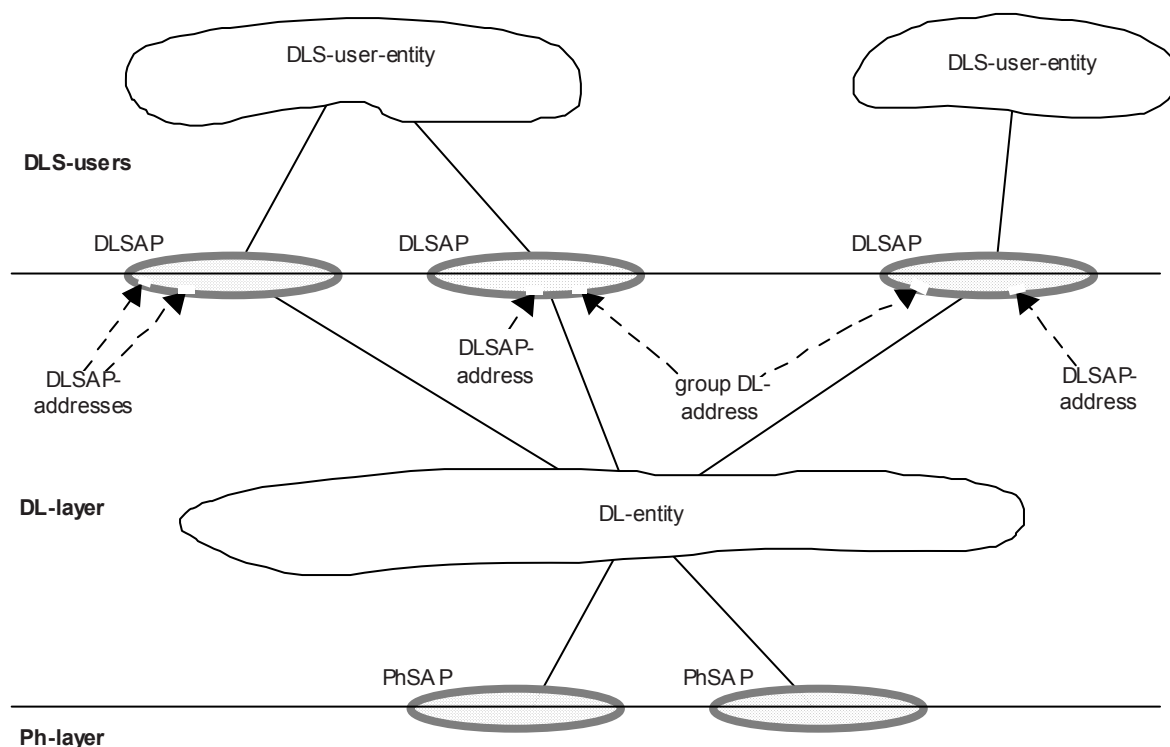
3.3 Common terms and definitions

NOTE Many definitions are common to more than one protocol Type; they are not necessarily used by all protocol Types.

3.3.1

DL-segment, link, local link

single DL-subnetwork in which any of the connected DLEs may communicate directly, without any intervening DL-relaying, whenever all of those DLEs that are participating in an instance of communication are simultaneously attentive to the DL-subnetwork during the period(s) of attempted communication



NOTE 1 DLSAPs and PhSAPs are depicted as ovals spanning the boundary between two adjacent layers.

NOTE 2 DL-addresses are depicted as designating small gaps (points of access) in the DLL portion of a DLSAP.

NOTE 3 A single DL-entity may have multiple DLSAP-addresses and group DL-addresses associated with a single DLSAP.

Figure 1 – Relationships of DLSAPs, DLSAP-addresses and group DL-addresses

3.3.2

DLSAP

distinctive point at which DL-services are provided by a single DL-entity to a single higher-layer entity.

NOTE This definition, derived from ISO/IEC 7498-1, is repeated here to facilitate understanding of the critical distinction between DLSAPs and their DL-addresses. (See Figure 1.)

3.3.3

DL(SAP)-address

either an individual DLSAP-address, designating a single DLSAP of a single DLS-user, or a group DL-address potentially designating multiple DLSAPs, each of a single DLS-user.

NOTE This terminology is chosen because ISO/IEC 7498-3 does not permit the use of the term DLSAP-address to designate more than a single DLSAP at a single DLS-user

3.3.4

(individual) DLSAP-address

DL-address that designates only one DLSAP within the extended link

NOTE A single DL-entity may have multiple DLSAP-addresses associated with a single DLSAP.

3.3.5

extended link

DL-subnetwork, consisting of the maximal set of links interconnected by DL-relays, sharing a single DL-name (DL-address) space, in which any of the connected DL-entities may communicate, one with another, either directly or with the assistance of one or more of those intervening DL-relay entities

NOTE An extended link may be composed of just a single link.

3.3.6

frame

denigrated synonym for DLPDU

3.3.7

group DL-address

DL-address that potentially designates more than one DLSAP within the extended link. A single DL-entity may have multiple group DL-addresses associated with a single DLSAP. A single DL-entity also may have a single group DL-address associated with more than one DLSAP

3.3.8

node

single DL-entity as it appears on one local link

3.3.9

receiving DLS-user

DL-service user that acts as a recipient of DL-user-data

NOTE A DL-service user can be concurrently both a sending and receiving DLS-user.

3.3.10

sending DLS-user

DL-service user that acts as a source of DL-user-data

3.4 Additional Type 2 definitions

3.4.1

actual packet interval (API)

measure of how frequently a specific connection produces its Lpacket data

3.4.2

allocate

take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.4.3

application

function or data structure for which data is consumed or produced

3.4.4

application objects

multiple object classes that manage and provide the run time exchange of messages across the network and within the device

3.4.5**attribute**

description of an externally visible characteristic or feature of an object

NOTE The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behavior of an object. Attributes are divided into class attributes and instance attributes.

3.4.6**behavior**

indication of how the object responds to particular events. Its description includes the relationship between attribute values and services

3.4.7**bit**

unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted

3.4.8**blanking or blanking time**

length of time required after transmitting before the node is allowed to receive

3.4.9**client**

- 1) An object which uses the services of another (server) object to perform a task
- 2) An initiator of a message to which a server reacts

3.4.10**communication objects**

components that manage and provide run time exchange of messages across the network such as the ControlNet object, the Unconnected Message Manager (UCMM)

3.4.11**connection**

logical binding between two application objects within the same or different devices

3.4.12**connection ID (CID)**

identifier assigned to a transmission, associated with a particular connection between producers and consumers, that identifies a specific piece of application information

NOTE Within the data link this is a generic tag.

3.4.13**cyclic redundancy check (CRC)**

residual value computed from an array of data and used as a representative signature for the array

3.4.14**cyclic**

term used to describe events which repeat in a regular and repetitive manner

3.4.15**deafness**

situation where the node can not hear the moderator DLPDU but can hear other link traffic

3.4.16**device**

physical hardware connection to the link

NOTE A device may contain more than one node.

3.4.17

DLPDU

data-link protocol data unit

NOTE A DLPDU consists of a source MAC ID, zero or more Lpackets and an FCS as transmitted or received by an associated PhE.

3.4.18

end delimiter

unique set of M_symbols that identifies the end of a DLPDU

3.4.19

error

discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.4.20

extended link

links connected by DL-routers (sometimes called a local area network)

3.4.21

FCS error

error that occurs when the computed frame check sequence value after reception of all the octets in a DLPDU does not match the expected residual

3.4.22

fixed tag

two octet identifier (tag) which identifies a specific service to be performed by either

- a) that receiving node on the local link which has a specified MAC ID, or
- b) all receiving nodes on the local link

NOTE Identification of the target node(s) is included in the two octet tag

3.4.23

frame

denigrated synonym for DLPDU

3.4.24

generic tag

three octet identifier (tag) which identifies a specific piece of application information

3.4.25

guardband

time slot allocated for the transmission of the moderator DLPDU

3.4.26

implicit token

mechanism that governs the right to transmit

NOTE No actual token message is transmitted on the medium. Each node keeps track of the MAC ID of the node that it believes currently holds the right to transmit. The right to transmit is passed from node to node by keeping a record of the node that last transmitted. A slot time is used to allow a missing node to be skipped in the rotation

3.4.27

implicit token register

register that contains the MAC ID of the node that holds the right to transmit

3.4.28**instance**

actual physical occurrence of an object within a class, identifying one of many objects within the same object class

EXAMPLE California is an instance of the object class state.

NOTE The terms object, instance, and object instance are used to refer to a specific instance.

3.4.29**instance attribute**

attribute that is unique to an object instance and not shared by the object class

3.4.30**instantiated**

object that has been created in a device

3.4.31**Keeper**

object responsible for distributing link configuration data to all nodes on the link

3.4.32**link**

collection of nodes with unique MAC IDs, attached to Ph-segments connected by Ph-repeaters

3.4.33**little endian**

model of memory organization which stores the least significant octet at the lowest address, and of transmission organization which transfers the least significant octet first on the medium

3.4.34**Lpacket**

well-defined sub-portion of a DLPDU consisting of

- a) size and control information
- b) a fixed tag or a generic tag, and
- c) DLS-user data or, when the tag has DL-significance, DL-data

3.4.35**M_symbol(s)**

symbols that represent the data bits and related information encoded and transmitted by the PhL

3.4.36**maximum scheduled node**

node with highest MAC ID that can use scheduled time on a link

3.4.37**maximum unscheduled node**

node with highest MAC ID that can use unscheduled time on a link

3.4.38**Message Router**

object within a node that distributes messaging requests to the appropriate application objects

3.4.39**moderator**

node with the lowest MAC ID that is responsible for transmitting the moderator DLPDU

3.4.40

moderator DLPDU

DLPDU transmitted by the node with the lowest MAC ID for the purpose of synchronizing the nodes and distributing the link configuration parameters

3.4.41

multipoint connection

connection from one node to many

NOTE Multipoint connections allow messages from a single producer (publisher) to be received by many consumer (subscriber) nodes.

3.4.42

network

series of nodes connected by some type of communication medium. The connection paths between any pair of nodes can include repeaters, routers and gateways

3.4.43

network access monitor (NAM)

component within a node that manages communication with a temporary node connected to the nodes local NAP

3.4.44

network access port (NAP)

PhL variant that allows a temporary node to be connected to the link by connection to the NAP of a permanent node

3.4.45

network address or node address

node's address on the link (also called MAC ID)

3.4.46

network status indicators

indicators on a node indicating the status of the Physical and Data Link Layers

3.4.47

network update time (NUT)

repetitive time interval in which data can be sent on the link

3.4.48

node

logical connection to a local link, requiring a single MAC ID

NOTE A single physical device may appear as many nodes on the same local link. For the purposes of this protocol, each node is considered to be a separate DLE.

3.4.49

non-concurrence

situation where a transmission is received from an unexpected MAC ID, which appears to violate the time based access protocol

NOTE This may occur when a connection is made between two working links that are not synchronized with each other but who have the same configuration information

3.4.50

non-data symbol

PhL symbol which violates the requirements of Manchester biphase L encoding

3.4.51

object

abstract representation of a particular component within a device

NOTE An object can be

- 1) an abstract representation of a computer's capabilities. Objects can be composed of any or all of the following components:
 - a) data (information which changes with time);
 - b) configuration (parameters for behavior);
 - c) methods (things that can be done using data and configuration).
- 2) a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior.

3.4.52

object specific service

service defined by a particular object class to perform a required function which is not performed by a common service

NOTE An object specific service is unique to the object class which defines it.

3.4.53

originator

client responsible for establishing a connection path to the target

3.4.54

permanent node

node whose connection to the network does not utilize the network access port (NAP) PhL variant

NOTE This node may optionally support a NAP PhL variant to allow temporary nodes to connect to the network

3.4.55

produce

act of sending data to be received by a consumer

3.4.56

producer

node that is responsible for sending data

3.4.57

redundant media

system using more than one medium to help prevent communication failures

3.4.58

requested packet interval (RPI)

measure of how frequently the originating application requires the transmission of Lpackets or data packets from the target application

3.4.59

rogue

node that has received a moderator DLPDU that disagrees with the link configuration currently used by this node

3.4.60

scheduled

data transfers that occur in a deterministic and repeatable manner on predefined NUTs

3.4.61

server

object which provides services to another (client) object

3.4.62

service

operation or function than an object and/or object class performs upon request from another object and/or object class

NOTE A set of common services is defined and provisions for the definition of object-specific services are provided. Object-specific services are those which are defined by a particular object class to perform a required function which is not performed by a common service.

3.4.63

slot time

maximum time required for detecting an expected transmission

NOTE Each node waits a slot time for each missing node during the implied token pass

3.4.64

start delimiter

unique set of M_symbols that identifies the beginning of a DLPDU

3.4.65

supernode

DLE with a MAC ID of zero

NOTE This DLE DL-address is reserved for special DLL functions

3.4.66

table unique identifier (TUI)

unique reference code used by ControlNet and Keeper objects to reference a consistent set of link configuration attribute

3.4.67

tag

shorthand name for a specific piece of application information, two or three octets in length

3.4.68

target

end-node to which a connection is established

3.4.69

temporary node

same as transient node

3.4.70

tMinus

number of NUTs before a new set of link configuration parameters are to be used

3.4.71

tone

instant of time which marks the boundary between two NUTs

3.4.72

tool

executable software program which interacts with the user to perform some function

EXAMPLE Link scheduling software (LSS).

3.4.73

transaction id

field within a UCMM header that matches a response with the associated request, which a server echoes in the response message

3.4.74**transient node**

node that is only intended to be connected to the network on a temporary basis using the NAP PhL medium connected to the NAP of a permanent node

3.4.75**Unconnected Message Manager (UCMM)**

component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.4.76**unconnected service**

messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

3.4.77**unscheduled**

data transfers that use the remaining allocated time in the NUT after the scheduled transfers have been completed

3.5 Type 2 symbols and abbreviations

ACM	Access control machine	
CA	Clock accuracy	
CAN	Controller area network	[ISO 11898:1993]
COP	Connection originator password	
IP	Internet Protocol	
LED	Light emitting diode.	
LSS	Link scheduling software	
MAC ID	MAC address of a node	
NAM	Network attachment monitor	
NAP	Network access port	
ND	Non-data symbol	
NUT	Network update time	
PT	Programming terminal (a temporary network connection)	
Rcv	Receive	
RPI	Requested Lpacket interval	
Rx	Receive	
RxLLC	Receive logical link control	
RxM	Receive machine	
SEM	State event matrix	
SMAX	MAC ID of the maximum scheduled node	
STD	State transition diagram, used to describe object behavior	
TCP	Terminal Control Protocol	
TUI	Table unique identifier	
Tx	Transmit	
TxLLC	Transmit logical link control	
TxM	Transmit machine	
UCCM	Unconnected Message Manager	
UMAX	MAC ID of maximum unscheduled node	
USR	Unscheduled start register	

4 Overview of the DL-protocol

4.1 General

4.1.1 DLL architecture

The Type 2 DLL is modeled as an integrated Access Control Machine (ACM), with scheduling support functions, designed for reliable and efficient support of higher-level connection-mode and connectionless data transfer services. Interoperating with these higher level functions are DLL management functions.

PhL framing and delimiters are managed by DLL functions for serializing and deserializing M_symbol requests and indications.

Within the Data Link Layer, the Access Control Machine (ACM) has the primary responsibility for detecting and recovering from network disruptions. The major objectives of the ACM are

- to make sure that the local node detects and fully utilizes its assigned access slots in the schedule;
- to make sure that the local node does not interfere with the transmissions of other nodes, especially the moderator node;
- to start the next NUT (see 4.1.2) on schedule, whether the moderator DLPDU is heard or not;
- if the local node is the moderator, to transmit each moderator DLPDU strictly on schedule.

The Data Link Layer is comprised of the components listed in Table 1 :

Table 1 – Data-link layer components

Components	Description
Access Control Machine (ACM)	receives and transmits control DLPDUs and header information, and determines the timing and duration of transmissions.
Transmit LLC (TxLLC)	buffers DLSDUs received from Station Management and the DLS-user, and determines which should be transmitted next.
Receive LLC (RxLLC)	performs the task of quarantining received link units until they are validated by a good FCS.
Transmit Machine (TxM)	receives requests to send DLPDU headers and trailers, and Lpackets from the ACM, and breaks them down into octet symbol requests that are sent to the Serializer.
Receive Machine (RxM)	assembles received Lpackets from octet symbols received from the Deserializer, and submits them to the RxLLC.
Serializer	receives octet symbols, encodes and serializes them, and sends them as M_symbols to the PhL. It is also responsible for generating the FCS.
Deserializer	receives M_symbols from the PhL, converts M_symbols into octets and sends them to the receive machine. It is also responsible for checking the FCS.
DLL Management Interface	holds the Station Management variables that belong to the DLL, and helps manage synchronized changes of the link parameters.

The internal arrangement of these components, and their interfaces, are shown in Figure 2. The arrowheads illustrate the primary direction of flow of data and control.

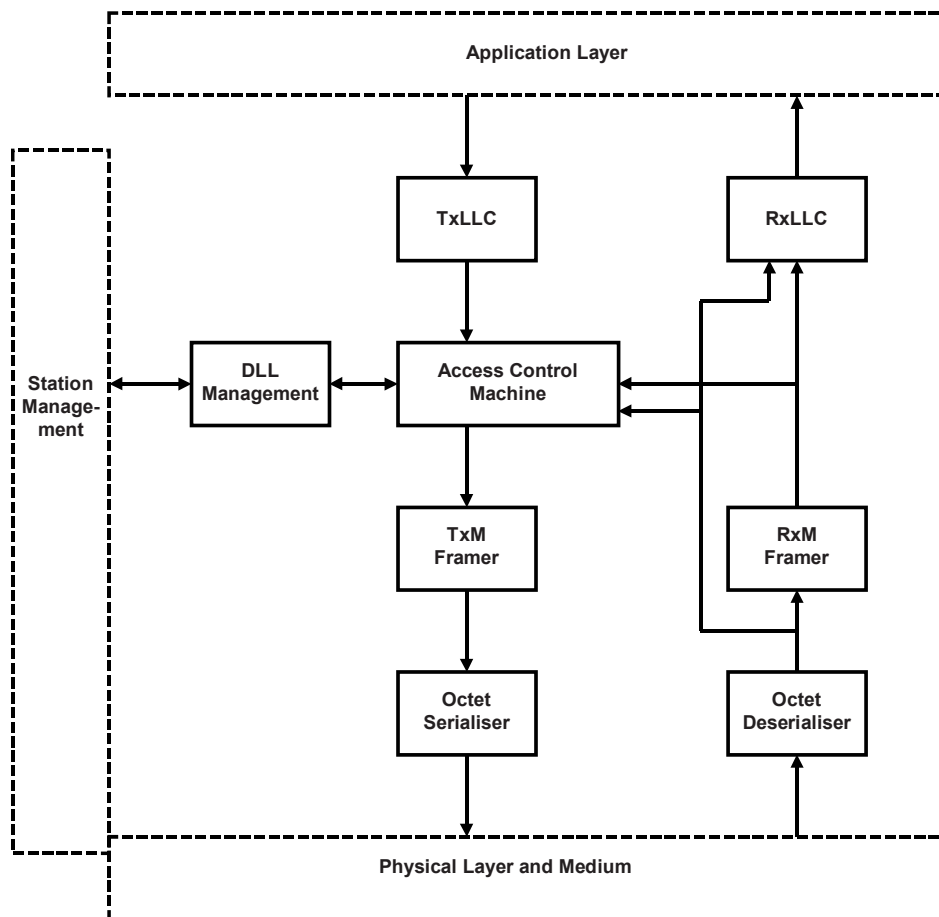


Figure 2 – Data-link layer internal architecture

4.1.2 Access control machine (ACM) and scheduling support functions

The ACM functions schedule all communication from one DLE to another. The timing of this communication are regulated at two levels,

- 1) to provide an accurate time based distribution of scheduled opportunities for designated communications in accord with, and with a pre-established schedule for, the local link and any extended network of links.
- 2) to provide a democratic distribution of unscheduled opportunities for arbitrary communications, generally in a cyclic but asynchronous manner.

Accurate schedule timing as provided by 1) is very important to support many control and data collection tasks in the applications domain of this protocol. The schedule is based on a fixed, repetitive time cycle called network update time (NUT). The NUT is maintained in close synchronism among all nodes participating in the DLL and used to provide two types of access opportunity;

- 1) One opportunity for each active station to transfer one scheduled DLPDU containing multiple scheduled DLSDUs.

NOTE 1 Only DLSDUs with scheduled as their QoS parameter may be included in a scheduled DLPDU.

NOTE 2 DLSDUs with scheduled as their QoS parameter are usually connection mode transfers with generic as their tag parameter.

- 2) One opportunity for at least one active station to send one background (unscheduled) DLPDU containing multiple DLSDUs of any QoS type. If time is available within a NUT, one background MAC opportunity is offered to each other active station in order of their MAC ID address. The station MAC ID for the first background opportunity in each NUT, is incremented (+1 modulo the set of background addresses) for each successive NUT to democratically share the available time among active stations.

Support procedures are included for automatic transfer to backup scheduling services and to manage the use of redundant Ph media.

4.1.3 Connection-mode, connectionless-mode data transfer and DL service

The connection mode, connectionless mode and DL services provide the necessary functionality for

- managing DL provider interactions with the DL user by converting request and response primitives into necessary DLE operations and generating indications and confirm primitives where appropriate,
- determination of the DL address and control information to be added to each DLSDU,
- local confirmation of status for each outgoing DLSDU,
- DLPDU formation by concatenation of multiple DL user DLSDUs for efficient transfer as single PhPDUs, subject to DLPDU size limitations, QoS parameters and the type of access opportunity.

4.2 Services provided by the DL

4.2.1 Overview

The DLL provides connectionless data transfer services and connection-mode data transfer services for limited-size DLSDUs, an internally synchronized time service, scheduling services to control the time allocation of the underlying shared PhL service, a DL(SAP)-address and queue management service, and a packing/unpacking service that combines multiple DLSDUs into a single data transfer DLPDU for efficient use of each access opportunity.

4.2.2 QoS

4.2.2.1 Definition

The QoS parameter specifies priority options for DLSDUs and access opportunities. The available values are as follows:

4.2.2.2 DLL scheduled priority

The scheduled priority QoS provides accurate time based cyclic and acyclic sending of DLSDUs. The execution timing for this fieldbus-scheduled service can be accurate and repeatable to better than 1 ms. Scheduled priority is normally used with connected mode services.

NOTE The master Keeper is responsible for regular publication of a unique table identifier (TUI) which ensures all nodes have a reference for the current set of link operating parameters. This TUI publication uses a fixed tag and is sent with QoS priority set to scheduled.

4.2.2.3 DLL high priority (unscheduled)

The high priority QoS provides acyclic sending of DLSDUs with a bounded upper time for the sending delay. Data on this priority is sent only when all scheduled data has been sent and a non-scheduled sending opportunity is available.

4.2.2.4 DLL low priority (unscheduled)

The low priority QoS provides for sending of DLSDUs only on an as-available basis. Data on this priority is sent only when all other data priorities have been sent and a non-scheduled sending opportunity is available.

NOTE High and Low priority are used only in a local sense to set the order of servicing among locally submitted, unscheduled DLS-user-data.

4.3 Structure and definition of DL-addresses

4.3.1 General

DL-addresses are used as MAC ID addresses (Link node designators), Fixed tag addresses (DLSAP-addresses and group DL-addresses), and Generic tag addresses (DLCEP-addresses).

This subclause defines the form and structure of DL-addresses and includes specific definitions for some standard DL-addresses.

All addresses are formed of octets. Each octet shall be transferred to the Ph layer low-bit first and multiple octets shall be transferred low-order octet first (little endian format).

Three types of DL addresses are used.

4.3.2 MAC ID address

MAC ID addresses identify physical nodes on the local link as shown in Figure 3.

Permitted values are within the range 0–99, values from 100–254 are reserved.



Figure 3 – Basic structure of a MAC ID address

Table 2 specifies the use of MAC ID addresses.

Table 2 – MAC ID addresses allocation

MAC ID	Meaning and purpose
0x0	temporary address used for maintenance
SMAX	current maximum address value for Scheduled access opportunities. SMAX ≤ UMAX
UMAX	current maximum address value for Unscheduled access opportunities SMAX. ≤ UMAX ≤ 0x63
0x64 to 0xFE	reserved
0xFF	broadcast address to all MAC ID nodes on this link

4.3.3 Generic tag address

Generic tag addresses types identify all data transfers for connected mode services. The available capacity in the DLL is 3 octets, see Figure 4.

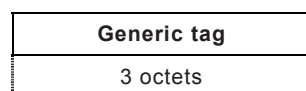


Figure 4 – Basic structure of a generic tag address

NOTE 1 Generic tags are used to send connected DLSDUs and may be associated with any of the available QoS priorities (Scheduled, High, Low).

NOTE 2 Negotiation and management of generic tags and connected mode services are the responsibility of the DL user.

4.3.4 Fixed tag address

Fixed tag addresses types identify DLSAPs within a designated station MAC ID. They are used with all data transfers for connectionless mode services.

NOTE Fixed tags are used to send connectionless DLSDUs and are usually associated with unscheduled QoS priorities (High, Low). Typical uses of fixed tags are for station management and to establish connections.

Each fixed tag address shall consist of two octets as shown in Figure 5.

Fixed tag service	Destination MAC ID
1 octet	1 octet

Figure 5 – Basic structure of a fixed tag address

The first octet shall specify the service access point in the destination node as specified in Table 3. The second octet shall be the address of the destination node. The destination address shall be either a MAC ID or 0xFF, which indicates the broadcast address to all MAC IDs on the local link.

Table 3 – Fixed tag service definitions

Fixed tag service	Meaning
0x00	moderator
0x01 - 0x08	vendor specific
0x09	ping request
0x0A – 0x14	vendor specific
0x15	tMinus
0x16 - 0x28	vendor specific
0x29	ping reply
0x2A - 0x3F	vendor specific
0x40 - 0x6F	reserved
0x70 – 0x7F	vendor-specific
0x80	I'm alive
0x81	link parameters
0x82	reserved
0x83	UCMM
0x84	TUI
0x85	IP (reserved for Internet Protocol)
0x86	WAMI
0x87	reserved
0x88	Keeper UCMM
0x89	Ethernet (used for address resolution protocol)
0x8A - 0x8B	reserved
0x8C	Time distribution
0x8D – 0x8F	reserved for time distribution
0x90	debug
0x91 - 0xCF	reserved
0xD0 - 0xEF	group addresses
0xF0 - 0xFF	vendor specific

Fixed tags in the range 0xD0 to 0xEF shall be reserved to permit group screening of connection identifiers.

4.4 Services assumed from the PhL

4.4.1 General requirements

The following subclauses include requirements for interface to a Type 2 PhL that are not required for other Types.

- Transmit Machine.
- Receive Machine.
- Serializer.
- Deserializer.

This is necessary because the Ph-Interface Data Units (PhIDUs) that Type 2 passes across the DLL–PhL interface are individual serial data and non-data symbols, designated on the DLE side as M_Symbols (see Table 4), whereas PhIDUs for other Types are not conceptualized in this way. As a consequence, a Type 2 Data Link Layer requires a Type 2 PhL.

4.4.2 Data encoding rules

The M_Symbols transferred at the DLL to PhL interface are of four types designated 0, 1, +, -. They will be encoded inside the PhL into appropriate Ph_Symbols as shown in Table 4. The M_0 and M_1 will be encoded into Ph_Symbols that represent Manchester biphas L data encoding rules. The M_ND symbols are non-data symbols to allow for the creation of unique data patterns used for start and end delimiters. The example of signal voltage waveform as it might be applied to an electrically-conductive medium is shown in an idealized form in Figure 6 to provide an example of the data encoding rules shown in Table 4.

Table 4 – Data encoding rules

Data bits (common name)	M_Symbol representation	Ph_Symbol encoding	Manchester encoded
data “zero”	M_0 or {0}	{L,H}	0
data “one”	M_1 or {1}	{H,L}	1
“non_data+”	M_ND+ or {+}	{H,H}	no data
“non_data-”	M_ND- or {-}	{L,L}	no data

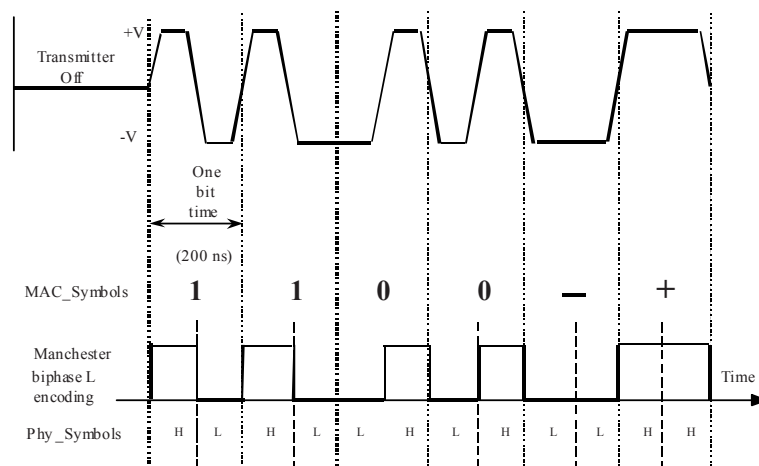


Figure 6 – M_symbols and Manchester encoding at 5 MHz (informative)

4.4.3 DLL to PhL interface

4.4.3.1 General

The DLL to PhL interface need not be exposed in the implementation of any PhL variant. This interface may be internal to the node. If conformance to the DLL to PhL interface is claimed, it shall conform to the requirements of this clause.

4.4.3.2 Ph_lock_indication

ph_lock_indication shall provide an indication of either data lock or Ph_Symbol synchronization by the PhL. Valid states for ph_lock_indication shall be true and false. ph_lock_indication shall be true whenever valid Ph_Symbols are present at the PhL interface and the PhL to DLL interface timing of M_Symbols conforms to the PhL requirements. It shall be false between frames (when no Ph_Symbols are present on the medium) or whenever data synchronization is lost or the timing fails to conform to the PhL requirements. ph_lock_indication shall be true prior to the beginning of the start delimiter.

4.4.3.3 Ph_frame_indication

ph_frame_indication shall provide an indication of a valid data frame from the PhL. Valid states for ph_frame_indication shall be true and false. ph_frame_indication shall be true upon ph_lock_indication = true and reception of the first valid start delimiter. ph_frame_indication shall be false at reception of next M_ND symbol (following the start delimiter) or ph_lock_indication = false.

NOTE This signal provides octet synchronization to the DLL.

4.4.3.4 Ph_carrier_indication

ph_carrier_indication shall represent the presence of a signal carrier on the medium. The ph_carrier_indication shall be true if internal PhL identification of signal carrier has been true during any of the last 4 M_symbol times and it shall be false otherwise.

4.4.3.5 Ph_data_indication

ph_data_indication shall represent the M_Symbols that are decoded from the PhL internal representations such as those shown in Table 4. Valid M_symbols shall be M_0 {0}, M_1 {1}, M_ND+ {+}, and M_ND- {-} as shown in Table 5.

Table 5 – M Data symbols

Data bits (common name)	M_Symbol representation
data “zero”	M_0 or {0}
data “one”	M_1 or {1}
“non_data+”	M_ND+ or {+}
“non_data-”	M_ND- or {-}

The ph_data_indication shall represent the M_Symbols as decoded within the PhL whenever ph_lock_indication is true.

4.4.3.6 Ph_status_indication

ph_status_indication shall represent the delimiter status of the frame which was received from the PhL as shown in Table 6. Valid symbols shall be Normal, Abort, and Invalid. ph_status_indication shall indicate Normal after reception of a frame (ph_frame_indication = true) composed of a start delimiter, valid data (no M_ND symbols) and an end delimiter. ph_status_indication shall indicate Abort after reception of a frame (ph_frame_indication = true) composed of a start delimiter, valid data, and a second start delimiter. ph_status_indication

shall indicate Invalid after reception of a frame (`ph_frame_indication = true`) composed of a start delimiter and the detection of any M_ND symbol which was not part of a start or end delimiter.

Table 6 – Truth table for `ph_status_indication`

<code>Ph_status_indication</code>	<code>Ph_frame_indication</code>	Start delimiters in a single frame	End delimiter detection	Any non-delimiter Manchester violations
Normal	true	1	true	false
Abort	true	2	don't care	false
Invalid	true	1	don't care	true

4.4.3.7 `Ph_data_request`

`ph_data_request` shall present the M_Symbols to be transmitted. Valid symbols shall be M_0, M_1, M_ND+ or M_ND– as shown in Table 5. `ph_data_request` shall indicate M_0 when no data is to be transmitted (and `ph_frame_request = false`).

4.4.3.8 `Ph_frame_request`

`ph_frame_request` shall be true when `ph_data_request` presents M_Symbols to be encoded into the appropriate Ph_Symbols by the PhL, and shall be false when no valid M_Symbols are to be transferred to the PhL.

4.5 Functional classes

Implementations of this protocol fall into two broad classes:

- connection originators (sometimes called Scanners);
- connection responders or connection targets (sometimes called Adaptors).

Devices with connection origination capability will include functions such as Scheduling object, support for external Configuration Tools and usually a Keeper object.

Connection responders are typically simpler devices which do not need the full set of capabilities.

This specification does not mandate particular groupings of functions for classes of implementation devices.

NOTE Grouping of services and functions into implementation classes may be done by separate specifications for profiles.

5 General structure and encoding of PhIDUs and DLPDUs and related elements of procedure

5.1 Overview

The DLL and its procedures are those necessary to provide the services offered to DLS user by using the services available from the PhL. The main services of this protocol are

- a) provision of timely access opportunities to support scheduled data-transfer DL-services on one link and on an extended multi-link network;
- b) the packing of multiple DLS-user transfers into a single PhL transfer unit (DLPDU) to efficiently use each transfer opportunity;
- c) the efficient distribution of DLS-user data from a publishing DLS-user to a set of subscribing DLS-users;
- d) the provision of a synchronized sense of internal time among the DLEs and available to the DLS-users of the extended link;
- e) the standardized availability of local DL-address, buffer and queue management capabilities.

5.2 Media access procedure

The primary task of the Data Link Layer is to determine, in co-operation with other Data Link Layers on the same link, the granting of permission to transmit on the medium. At its upper interface, the DLL provides services to receive and deliver service data units for the DLS user and Station Management.

The DLL protocol is based on a fixed, repetitive time cycle called a network update time (NUT). The NUT is maintained in close synchronism among all nodes on the link. A node is not permitted access to transmit on the medium if its NUT does not agree with the NUT currently being used on a link wide basis. Different links may have different NUT duration.

Each node contains its own timer synchronized to the NUT for the local link. Media access is determined by local sub-division of the NUT into access slots. Access to the medium is in sequential order based on the MAC ID of the node. Specific behaviors have been incorporated into the access protocol allowing a node which temporarily assumes a MAC ID of zero to perform link maintenance. The MAC ID numbers of all nodes on a link are unique. Any DLL detecting the presence of a duplicate MAC ID shall immediately stop transmitting.

An implicit token passing mechanism is used to grant access to the medium. Each node monitors the source MAC ID of each DLPDU received. At the end of a DLPDU, each DLL sets an "implicit token register" to the received source MAC ID + 1. If the implicit token register is equal to the local MAC ID, then the node shall transmit one DLPDU containing one or more Lpackets with data or a null DLPDU. In all other cases, the node watches for either a new DLPDU from the node identified by the "implicit token register" or a time-out value if the identified node fails to transmit. In each case, the "implicit token" is automatically advanced to the next MAC ID. All nodes have the same value in their "implicit token register" preventing collisions on the medium.

The time-out period (called the "slot time") is based on the amount of time required for

- the current node to hear the end of the transmission from the previous node;
- the current node to begin transmitting;
- the next node to hear the beginning of the transmission from the current node.

The slot time is adjusted to compensate for the total length of the medium since the propagation delay of the medium effects the first and last item on the previous list.

NOTE The calculation of slot time is specified in 8.2.

Each NUT is divided into three major parts: scheduled, unscheduled, and guardband as shown in Figure 7. This sequence is repeated in every NUT. The implicit token passing mechanism is used to grant access to the medium during both the unscheduled and scheduled intervals.

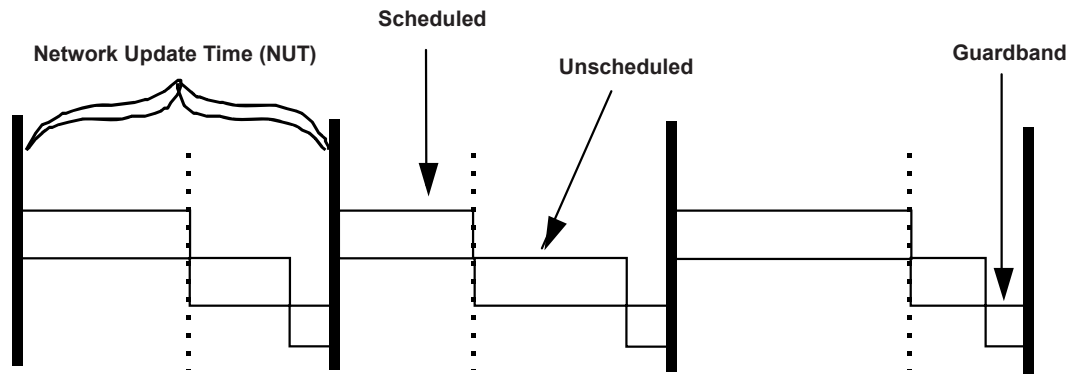


Figure 7 – NUT structure

During the scheduled part of the NUT, each node, starting with node 0 and ending with node SMAX, gets a chance to transmit time-critical (scheduled) data. SMAX is the MAC ID of the highest numbered node that has access to the medium during the scheduled part of the NUT. Every node between 0 and SMAX has only one opportunity to send one DLPDU of scheduled data in each NUT. The opportunity to access the medium during the scheduled time is the same for each node in every NUT. This allows data that is transmitted during the scheduled portion of the NUT to be sent in a predictable and deterministic manner. Figure 8 shows how the permission to transmit is granted during the scheduled time. The DLS-user regulates the amount of data that each node may transmit during this scheduled token pass.

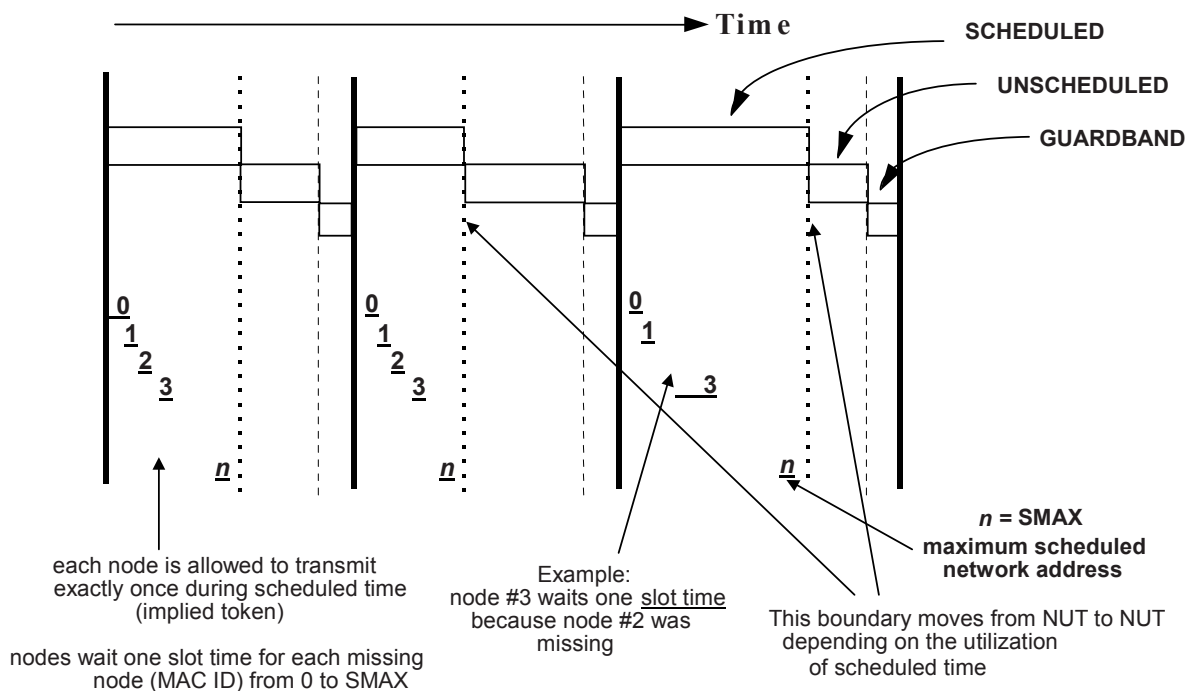


Figure 8 – Media access during scheduled time

During the unscheduled part of the NUT, each node from 0 to UMAX shares the opportunity to transmit one DLPDU of non-time critical data in a round robin fashion, until the allocated NUT duration is exhausted. UMAX is the MAC ID of the highest numbered node that has access to the medium during the unscheduled part of the NUT.

the medium during the unscheduled part of the NUT. The round robin method of access opportunity enables every node between 0 and UMAX to have zero, one or many opportunities to send unscheduled data depending on how much of the NUT remains after the completion of the scheduled time. Variations in scheduled traffic means the opportunity to access the medium during the unscheduled time may be different for each node in every NUT. Figure 9 shows how the permission to transmit is granted during the unscheduled time. The MAC ID of the node that goes first in the unscheduled part of the NUT is incremented by 1 for each NUT. The unscheduled token begins at the MAC ID specified in the unscheduled start register (USR) of the previous moderator DLPDU. The USR increments by one modulo (UMAX+1) each NUT. If the USR reaches UMAX before the guardband, it returns to zero and the token pass continues.

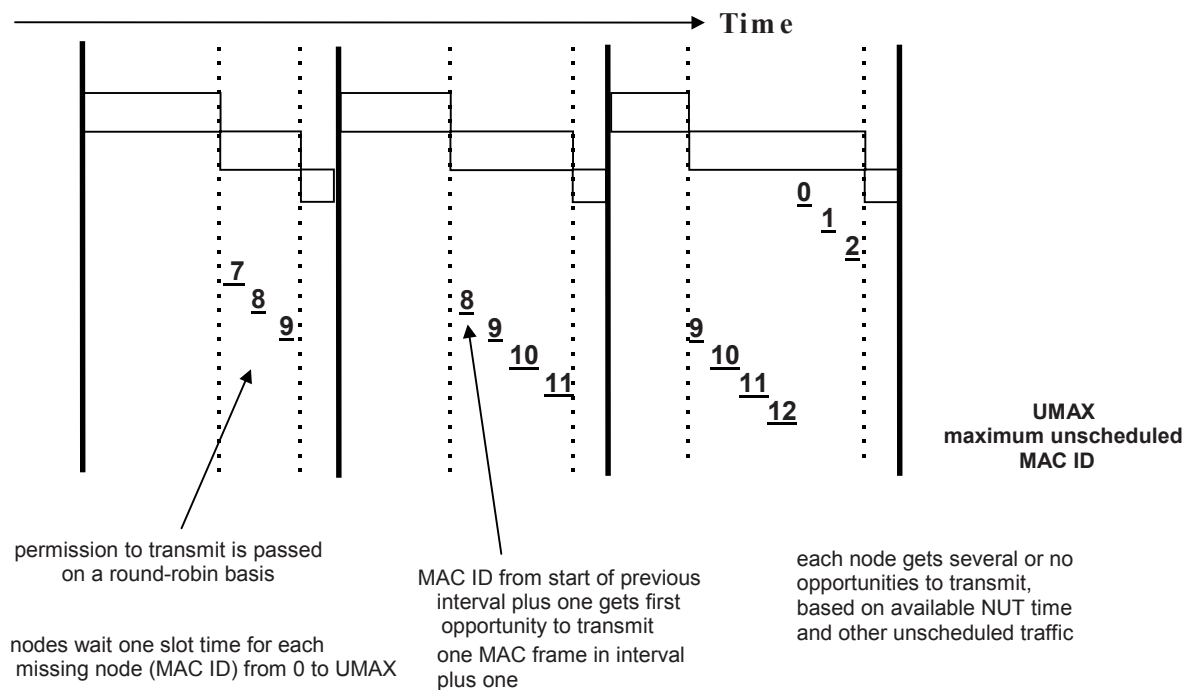


Figure 9 – Media access during unscheduled time

When the guardband is reached, all nodes stop transmitting. A node is not allowed to start a transmission unless it can be completed before the beginning of the guardband. During the guardband, the node with the lowest MAC ID (called the “moderator”) transmits a maintenance message (called the “moderator DLPDU”) that accomplishes two things:

- it keeps the NUT timers of all nodes synchronized;
- it publishes critical link parameters enabling all members of the local DLL group to share a common version of important DLL values such as NUT, slot time, SMAX, UMAX, etc.

The moderator transmits the moderator DLPDU, which re-synchronizes all nodes and restarts the NUT. Following the receipt of a valid moderator DLPDU, each node compares its internal values with those transmitted in the moderator DLPDU. A node using link parameters that disagree with the moderator disables itself. If the moderator DLPDU is not heard for 2 consecutive NUTs, the node with the lowest MAC ID assumes the moderator role, and begins transmitting the moderator DLPDU in the guardband of the third NUT. A moderator node that notices another node on-line and transmitting with a MAC ID lower than its own immediately cancels its moderator role.

Typical situations that may cause disruption of the DLL access protocol include

- induced noise on the link;
- poor cable or termination quality;

— physically connecting two links together while the network is operating.

One common consequence of such disruption is that nodes may be caused to disagree as to which node should be transmitting; this is called a network “non-concurrence”. Another potential problem occurs when the nodes do not agree to the same link configuration parameters. A node that disagrees with the link parameters as transmitted by the moderator is called a “rogue” and immediately stops transmitting. The DLL access protocol is designed to recover a rogue node and bring it back on-line.

5.3 DLPDU structure and encoding

5.3.1 General

DLPDUs are the DLPDUs passed to the Ph layer for sending on the Ph medium. The DLPDUs are formed by the transmit logical link control (TxLLC) protocol in two stages:

- DLSDUs are formed into Link-units (Lpackets);
- Multiple Lpackets are packed into one DLPDU subject to constraints set by the QoS required, the type of access opportunity available and limits on DLPDU overall length.

5.3.2 DLPDU components

The general DLPDU structure is shown in Figure 10 and managed by the DLL support units TxFramer, RxFramer, Octet Serializer and Octet Deserializer as shown in Figure 2.

The components of the DLPDU shall be transmitted in the following order: preamble, start delimiter, source MAC ID, zero or more Lpackets, FCS and end delimiter.

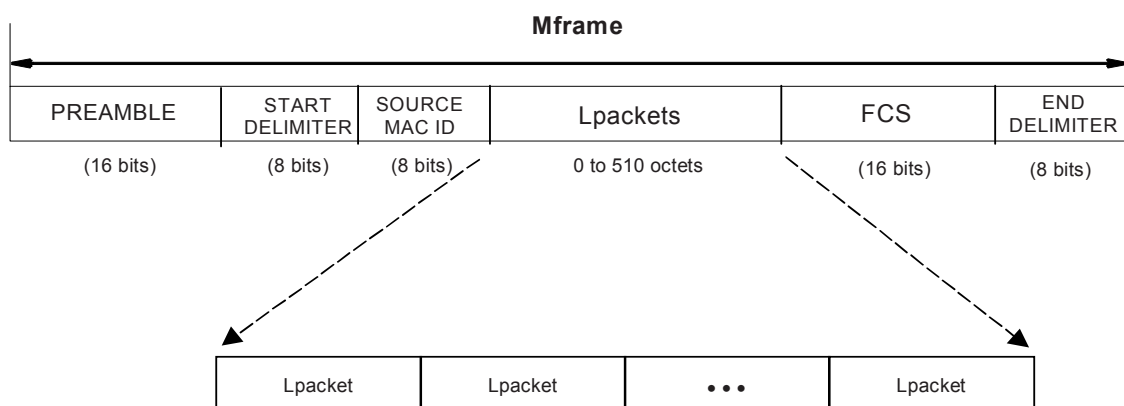


Figure 10 – DLPDU format

5.3.3 Preamble

The preamble shall be composed of 16 consecutive {1} M_Symbols.

5.3.4 Start and end delimiters

The start delimiter shall consist of these M_Symbols, {+, 0, –, +, –, 1, 0, 1}, transmitted from left to right.

The end delimiter shall consist of these M_Symbols, {1, 0, 0, 1, +, –, +, –}, transmitted from left to right.

The use of non-data M_Symbols shall be reserved for the start and end delimiters.

5.3.5 DLPDU octets and ordering

The source MAC ID, Lpackets, and FCS shall be composed of octets. An octet shall be composed of exactly eight M_Symbols each of which shall be either {0} or {1}. An octet shall be transferred to the PhL low bit first. Multiple octet fields shall be transmitted low order octet first (little endian format).

5.3.6 Source MAC ID

The source MAC ID address shall be a single octet as specified in 4.3.2. A node may also temporarily assume a MAC ID of zero to perform link maintenance.

5.3.7 Lpackets field

Zero or more Lpackets shall be concatenated into a single DLPDU subject to the constraint that the maximum number of octets which compose all Lpackets in a DLPDU shall be 510.

An additional constraint is that DLPDUs sent during a scheduled access opportunity shall only contain Lpackets with a QoS parameter of scheduled.

5.3.8 Frame check sequence (FCS)

5.3.8.1 Frame check sequence requirements

The FCS shall be a modified CCITT checksum using the 16 bit polynomial: $G(X) = X^{16} + X^{12} + X^5 + 1$. A two octet FCS shall be included in each DLPDU.

The formal concepts for its generation and checking on reception are described in 5.3.8.2. The generation and checking of the FCS are further specified in 9.7 and 9.8.

5.3.8.2 Frame check sequence formal concepts

5.3.8.2.1 Frame check sequence field

Within this subclause, any reference to bit K of an octet is a reference to the bit whose weight in a one-octet unsigned integer is 2^K .

NOTE This is sometimes referred to as "little endian" bit numbering.

For this standard, as in other International Standards (for example, ISO/IEC 3309, ISO/IEC 8802 and ISO/IEC 9314-2), DLPDU-level error detection is provided by calculating and appending a multi-bit frame check sequence (FCS) to the other DLPDU fields during transmission to form a "systematic code word"³ of length n consisting of k DLPDU message bits followed by $n - k$ (equal to 16) redundant bits, and by calculating during reception that the message and concatenated FCS form a legal (n, k) code word. The mechanism for this checking is as follows, where the generic form of the generator polynomial for this FCS construction is specified in equation (4) and the polynomial for the receiver's expected residue is specified in equation (9). The specific polynomials for this standard are specified in Table 7.

³ W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes* (2nd edition), MIT Press, Cambridge, 1972.

Table 7 – FCS length, polynomials and constants

Item	Value
$n-k$	16
$G(X)$	$X^{16} + X^{12} + X^5 + 1$ (notes 1, 2)
$R(X)$	$X^{12} + X^{11} + X^{10} + X^8 + X^3 + X^2 + X + 1$ (notes 2, 3)
NOTE 1 Code words $D(X)$ constructed from this $G(X)$ polynomial have Hamming distance 4 for lengths $\leq 4\,095$ octets, and all errors of odd weight are detected.	
NOTE 2 These are the same polynomials and method as specified in ISO/IEC 3309 (HDLC).	
NOTE 3 The remainder $R(x)$ should be 0001 1101 0000 1111 (X^{15} to X^0 , respectively) in the absence of errors.	

5.3.8.2.2 At the sending DLE

The original message (that is, the DLPDU without an FCS), the FCS, and the composite message code word (the concatenated DLPDU and FCS) shall be regarded as vectors $M(X)$, $F(X)$, and $D(X)$, of dimension k , $n - k$, and n , respectively, in an extension field over $GF(2)$. If the message bits are $m_1 \dots m_k$ and the FCS bits are $f_{n-k-1} \dots f_0$, where

- $m_1 \dots m_8$ form the first octet sent,
- $m_{8N-7} \dots m_{8N}$ form the N th octet sent,
- $f_7 \dots f_0$ form the last octet sent, and
- m_1 is sent by the first PhL symbol(s) of the message and f_0 is sent by the last PhL symbol(s) of the message (not counting PhL framing information),

NOTE 1 This “as transmitted” ordering is critical to the error detection properties of the FCS.

then the message vector $M(X)$ shall be regarded to be

$$M(X) = m_1X^{k-1} + m_2X^{k-2} + \dots + m_{k-1}X^1 + m_k \quad (1)$$

and the FCS vector $F(X)$ shall be regarded to be

$$\begin{aligned} F(X) &= f_{n-k-1}X^{n-k-1} + \dots + f_0 \\ &= f_{15}X^{15} + \dots + f_0 \end{aligned} \quad (2)$$

The composite vector $D(X)$, for the complete DLPDU, shall be constructed as the concatenation of the message and FCS vectors

$$\begin{aligned} D(X) &= M(X)X^{n-k} + F(X) \\ &= m_1X^{n-1} + m_2X^{n-2} + \dots + m_kX^{n-k} + f_{n-k-1}X^{n-k-1} + \dots + f_0 \\ &= m_1X^{n-1} + m_2X^{n-2} + \dots + m_kX^{16} + f_{15}X^{15} + \dots + f_0 \quad (\text{for the case of } k = 15) \end{aligned} \quad (3)$$

The DLPDU presented to the PhL shall consist of an octet sequence in the specified order.

The redundant check bits $f_{n-k-1} \dots f_0$ of the FCS shall be the coefficients of the remainder $F(X)$, after division by $G(X)$, of $L(X)(X^k + 1) + M(X)X^{n-k}$

where $G(X)$ is the degree $n-k$ generator polynomial for the code words

$$G(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + \dots + 1 \quad (4)$$

and $L(X)$ is the maximal weight (all ones) polynomial of degree $n-k-1$

$$\begin{aligned} L(X) &= (X^{n-k} + 1) / (X + 1) = X^{n-k-1} + X^{n-k-2} + \dots + X + 1 \\ &= X^{15} + X^{14} + X^{13} + X^{12} + \dots + X^2 + X + 1 \quad (\text{for the case of } k = 15) \end{aligned} \quad (5)$$

That is,

$$F(X) = L(X) (X^k + 1) + M(X) X^{n-k} \text{ (modulo } G(X)) \quad (6)$$

NOTE 2 The $L(X)$ terms are included in the computation to detect initial or terminal message truncation or extension by adding a length-dependent factor to the FCS.

NOTE 3 As a typical implementation when $n-k = 16$, the initial remainder of the division is preset to all ones. The transmitted message bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial $G(X)$, specified in equation (7). The ones complement of the resulting remainder is transmitted as the $(n-k)$ -bit FCS, with the coefficient of X^{n-k-1} transmitted first.

The equation for $D(X)$ does not apply to the Type 8 protocol because that protocol sends the message and check bits in separate DLPDUs. The equation for $F(X)$ as just given does not apply to the Type 8 protocol because that protocol does not complement the check bits before transmission. The equation corresponding to $F(X)$ for the Type 8 protocol is

$$F_{\text{Type 8}}(X) = L(X) X^k + M(X) X^{n-k} \text{ (modulo } G(X)) \quad (7)$$

NOTE 4 As a typical implementation for the Type 8 protocol, the initial remainder of the division is preset to all ones. The transmitted message bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial $G(X)$, specified in equation (7). The resulting remainder without ones complementation is transmitted as the $(n-k)$ -bit FCS, with the coefficient of X^0 transmitted first.

5.3.8.2.3 At the receiving DLE

NOTE 1 This subclause does not apply to the Type 8 protocol.

The octet sequence indicated by the PhE shall be concatenated into the received DLPDU and FCS, and regarded as a vector $V(X)$ of dimension u

$$V(X) = v_1 X^{u-1} + v_2 X^{u-2} + \dots + v_{u-1} X + v_u \quad (8)$$

NOTE 2 Because of errors u can be different than n , the dimension of the transmitted code vector.

A remainder $R(X)$ shall be computed for $V(X)$, the received DLPDU and FCS, by a method similar to that used by the sending DLE (see 5.3.8.2.2) in computing $F(X)$

$$\begin{aligned} R(X) &= L(X) X^u + V(X) X^{n-k} \text{ (modulo } G(X)) \\ &= r_{n-k-1} X^{n-k-1} + \dots + r_0 \end{aligned} \quad (9)$$

Define $E(X)$ to be the error code vector of the additive (modulo-2) differences between the transmitted code vector $D(X)$ and the received vector $V(X)$ resulting from errors encountered (in the PhS provider and in bridges) between sending and receiving DLEs.

$$E(X) = D(X) + V(X) \quad (10)$$

If no error has occurred, so that $E(X) = 0$, then $R(X)$ will equal a non-zero constant remainder polynomial

$$R_{0k}(X) = L(X) X^{n-k} \text{ (modulo } G(X)) \quad (11)$$

whose value is independent of $D(X)$. Unfortunately $R(X)$ will also equal $R_{0k}(X)$ in those cases where $E(X)$ is an exact non-zero multiple of $G(X)$, in which case there are “undetectable” errors. In all other cases, $R(X)$ will not equal $R_{0k}(X)$; such DLPDUs are erroneous and shall be discarded without further analysis.

NOTE 3 As a typical implementation, the initial remainder of the division is preset to all ones. The received bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial $G(X)$, specified in equation (8).

5.3.9 Null DLPDU

A DLPDU with an empty Lpacket field, called a null DLPDU, shall be sent by a node to indicate that it has no data to send at the QoS allowed by the access opportunity.

5.3.10 Abort DLPDU

To abort a partially transmitted DLPDU, the Data Link Layer shall transmit the following sequence:

- one 0xFF octet;
- start delimiter;
- end delimiter.

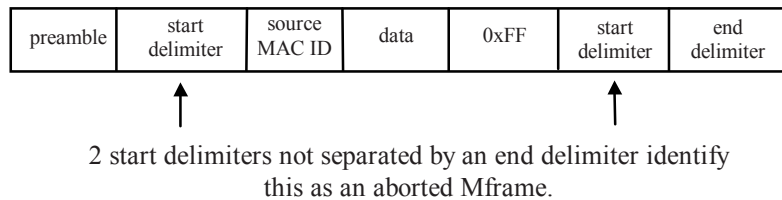


Figure 11 – Aborting a DLPDU during transmission

NOTE 1 Some implementations of this specification may build the DLPDU as it is being transmitted. If a higher layer has provided only a partial DLPDU to the Data Link Layer, and the DLL has begun transmission of the DLPDU assuming that the rest will be provided shortly, a transmit underflow occurs if the rest of the DLPDU is not provided to the DLL soon enough to complete the transmission.

NOTE 2 Although the DLL assembles the entire DLPDU including delimiters, the PhL performs start and end delimiter detection. The PhL identifies a transmit abort as a DLPDU which includes two start delimiters which are not separated by an end delimiter, as shown in Figure 11.

NOTE 3 The 0xFF octet is inserted before the start delimiter / end delimiter pair to avoid violating PhL constraints governing consecutive physical symbol relationships on the media.

5.4 Lpacket components

5.4.1 General Lpacket structure

The Lpacket (or link unit) is the PDU which is used to carry DLSDUs between peer DLS users. The components of the Lpacket shall be transmitted in the following order: size, control, tag, and link data as shown in Figure 12.

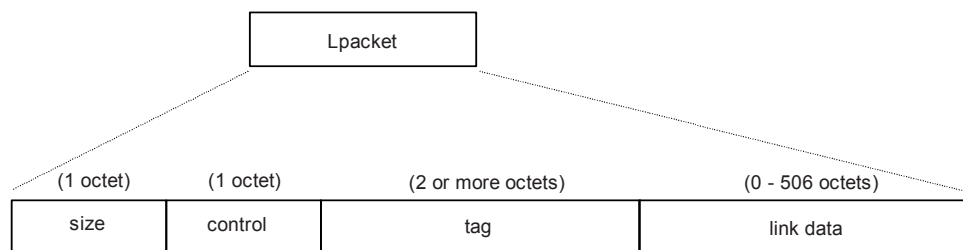


Figure 12 – Lpacket format

5.4.2 Size

The size field shall specify the number of octet pairs (from 3 to 255) contained in an individual Lpacket. The number of octet pairs shall include the size, control, tag, and link data fields counted in accordance with the following rules:

- combined, the size and control shall count as a single octet pair;
- the number of octet pairs from the tag and link data fields shall be counted separately and odd octet counts shall be rounded up;
- although the counting of octet pairs is done independently, the format of an Lpacket as placed into a DLPDU shall not include the extra octets caused by the round up.

NOTE For example, an Lpacket with a 3 octet tag (2 octet pairs) and a 9 octet link data (5 octet pairs) has a size field of $1+2+5 = 8$ octet pairs; however the Lpacket occupies $2+3+9 = 14$ octets in the DLPDU.

5.4.3 Control

5.4.3.1 Bit 0 and Bit 4 (type of Lpacket)

The bits of the control field shall be numbered 0 to 7, where bit 0 shall be the least significant bit of the control field. Bit 0 shall indicate the type of Lpacket.

- When set (bit 0 = 1), the Lpacket shall be a fixed tag Lpacket (see 4.3.4).
- When clear (bit 0 = 0), the Lpacket shall be a generic tag Lpacket (see 4.3.3).

Bit 4 of the control field shall be the inverse of bit 0. If bit 0 is clear, then bit 4 shall be set. If bit 0 is set, then bit 4 shall be clear.

5.4.3.2 Bit 1 (odd tag size)

Bit 1 of the control field shall indicate whether the tag field contains an even or odd number of octets. When clear (bit 1 = 0), this bit shall indicate that the tag contains an even number of octets. When set (bit 1 = 1), it shall indicate that the tag contains an odd number of octets.

5.4.3.3 Bit 2 (odd link data size)

Bit 2 of the control field shall indicate whether the link data contains an even or odd number of octets. When clear (bit 2 = 0), this bit shall indicate that the link data contains an even number of octets. When set (bit 2 = 1), it shall indicate that the link data contains an odd number of octets.

5.4.3.4 Bits 3, 5, 6 and 7 (reserved bits)

Bits 3, 5, 6 and 7 of the control field are reserved and shall be zero.

5.4.4 Generic tag Lpackets

Generic link-units are formed from the DLS-generic-tag and DLS-user-data provided by the DLS user with the addition of size and control field information for generic tags as shown in Figure 13. The 3 octet generic tag address (see 4.3.3) shall identify the connected mode information carried in the link data field

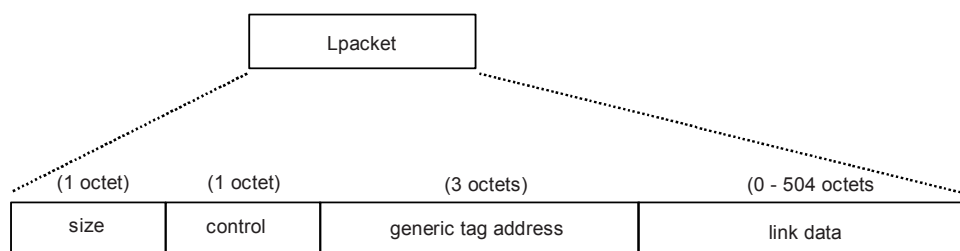


Figure 13 – Generic tag Lpacket format

The minimum length generic tag Lpacket shall be 5 octets (minimum link data size of 0 octets). The maximum size of link data shall be 504 octets.

NOTE Generic Lpackets are usually connection mode transfers with the QoS value of scheduled or low.

5.4.5 Fixed tag Lpackets

Fixed tag link-units are formed from the DLS-fixed-tag and DLS-user-data provided by the DLS user with the addition of size and control field information for fixed tags as shown in Figure 14. The 2 octet fixed tag address (see 4.3.4) shall identify the service access point in the the destination node and the address of the destination node.

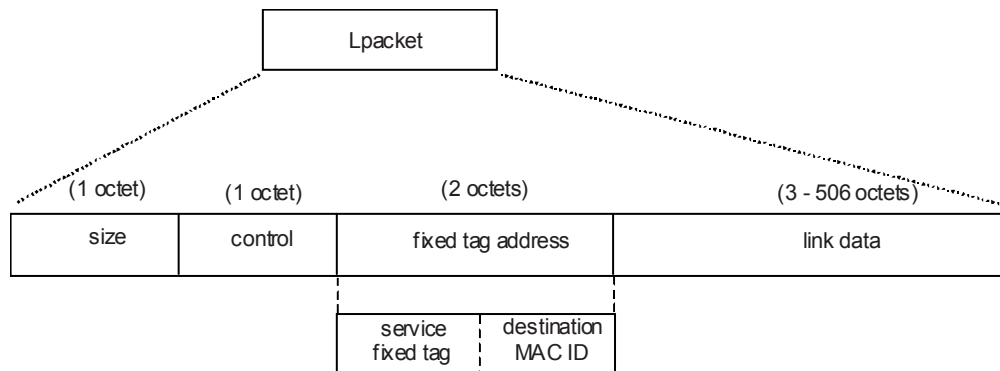


Figure 14 – Fixed tag Lpacket format

The minimum length fixed tag Lpacket shall be 7 octets (minimum link data size of 3 octets). The maximum size of link data shall be 506 octets.

NOTE Fixed Lpackets are always connectionless mode transfers. They are not usually sent with the QoS parameter value scheduled.

5.5 DLPDU procedures

5.5.1 General

The transmission protocol combines one or more Lpackets into single DLPDUs for sending according to the available transmission opportunity. The DLPDU components are passed to the Ph layer by the transmit machine framer (TxM) and Serializer (see 9.3).

A node shall always send one DLPDU at each valid access opportunity for it to transmit, and no DLPDU shall include Lpackets with a lesser QoS parameter than the QoS value applicable to the access opportunity.

If a node has no Lpackets available at or better than the QoS of the access opportunity, then a Null DLPDU shall be sent.

If for internal reasons a node begins a DLPDU and is unable to complete it, the abort sequence shall be sent, transmission shall stop for that access opportunity and the local DLS user shall be notified.

NOTE Null DLPDUs and abort DLPDUs provide an efficiency improvement as they notify other nodes that use of the transmission slot has ended and enable the next node to begin transmission without waiting for slot time out.

Each Lpacket in the queue awaiting transfer has an associated local identifier. At completion of the transfer to the PhL, or an aborted transfer, the local identifier is returned to the local DLS user with a status confirmation of OK or TXABORT.

The DLS user has additional local services to identify unsent Lpackets and request they are flushed from the queue, this action results in the confirmation status of FLUSHED being returned with the local identifier for Lpackets that have been flushed.

5.5.2 Sending scheduled DLPDUs

Scheduled DLPDUs are normally used for transfer of connected mode operation using Generic Lpackets. They are formed by concatenating multiple Lpackets (subject to a limit of 510 octets), prepending Ph-control information and source station MAC ID, and appending a defined frame check sequence and further Ph-control information.

Lpackets are packed into a Scheduled DLPDU in FIFO order as received from the DLS user, until the queue is empty of packets with Scheduled QoS or the next such Lpacket size exceeds the remaining space.

5.5.3 Sending unscheduled DLPDUs

Unscheduled DLPDUs may contain both Generic Lpackets and Fixed Lpackets. They are formed by concatenating multiple Lpackets (subject to a limit of 510 octets), prepending Ph-control information and source station MAC ID, and appending a defined frame check sequence and further Ph-control information.

Lpackets are packed into an unscheduled DLPDU in QoS priority order (Scheduled, High and Low priorities respectively). Within each QoS, the FIFO order of queuing by the DLS user is followed, until the queue is empty or the next such Lpacket size exceeds the remaining space.

5.5.4 Receiving DLPDUs

5.5.4.1 Unpacking

Received DLPDUs are assembled from the incoming Ph stream of symbols and unpacked by the Octet Deserializer, Receive Machine Framer (RxM) and Receive Logical Link Control (RxLLC) functions in two stages

- a frame check sequence is computed over the stream of received octets, (excluding Ph delimiters) and checked for the proper residual value, if this is correct, then
- each Lpacket contained in the DLPDU is examined to extract its tag. Each extracted tag is then compared with the contents of a local tag filter to determine if the associated Lpacket is of interest to the receiving station. Lpackets whose tags match the tag filter are then processed further and passed to the DL user, DL management or used inside the DLL. Indications passed to the DL user or DL management include Lpacket size, Lpacket tag and Lpacket data. In addition for fixed tag Lpackets, the source ID is also indicated.

5.5.4.2 Tag filtering

Each local DLS provider maintains a list of tags used to identify Lpackets of interest to its DL user or its internal DL management functions. The DL user may access and change this list to specify

- generic tags of interest
- fixed tags of interest

NOTE A generic tag specifies the Lpackets for a single connection-mode object. A fixed tag specifies all connectionless-mode messages in a particular class, for example Time distribution.

5.6 Summary of DLL support services and objects

The operation and coordination of participating DLL providers is governed by variables and attributes held in DL management objects or provided by the DL user services listed in Table 8. Access to these items is via defined fixed tag addresses, DL user services or unspecified local services. Details are given in the listed clauses.

Table 8 – DLL support services and objects

DLL item	Description	Location
DLS user services		see IEC 61158-3-2
connected mode transfer service	a local service for requesting, indicating and confirming generic tag (connected mode) data transfer by Lpackets	see 6.2.2
connectionless mode transfer service	a local service for requesting, indicating and confirming generic tag (connected mode) data transfer by Lpackets	see 6.2.2
queue maintenance service	a local service for flushing unsent Lpacket messages from the internal DL service queue	see 6.2.3
tag filter service	a local service for the DLS user to inform its local DL provider which Lpacket tags and tag types are to be accepted and processed by the DLL or passed to the DL user	see 6.2.4
link synchronization service	a local service for informing the local DLS user of the current NUT number to maintain schedule pointers	see 6.2.5
synchronized parameter change service and tMinus-start-countdown	a local service for loading and changing local lists of current and pending DL configuration data and in conjunction with the tMinus-start-countdown service and moderator DLPDU, to manage an accurately timed change over to new DL link configurations	see 6.2.6
event reports service	local services to report various events for use by management	see 6.2.7
bad FCS service		see 6.2.8
current moderator		see 6.2.9
power-up and on-line		see 6.2.10
enable moderator		see 6.2.11
listen only		see 6.2.12
Generic tag service	an Lpacket service for connected mode communication	see 6.3
Fixed tag services	a range of Lpacket service addresses for various connectionless mode communication	see 5.4.5
moderator	an Lpacket broadcast in every NUT for managing time precision and timing for change of link parameters	see 6.4
time distribution service	an Lpacket for publishing time values and offsets used in coordinating the same time sense in all stations	see 6.5
UCMM	an Lpacket service to transfer DLSDUs from a DLS user to the Unconnected Message Manager in a peer DLS-user	see 6.6
Keeper UCMM	an Lpacket service to transfer DLSDUs from a DLS user to the Keeper objects in other nodes	see 6.7
TUI	an Lpacket broadcast to all stations assisting their synchronization to the current link configuration	see 6.8
link parameters	an Lpacket service for publishing pending link parameters in advance of a synchronized change of link operating conditions	see 6.9
tMinus	an Lpacket service to start a count down leading to a change of link parameters and return the result.	see 6.9
I'm alive	an Lpacket broadcast by new and reset devices to assist their entry into an operating system	see 6.10
ping request and reply	an Lpacket service soliciting identification responses from all other active devices on the link	see 6.11
WAMI	an Lpacket allowing temporary devices to identify a local device MAC ID when they are plugged into its network access port	see 6.12
debug	an Lpacket service to trace object state transitions	see 6.13

DLL item	Description	Location
Objects		
ControlNet object	holds and distributes station management data classes and instances for Ph and DL layers, also collects diagnostic information for use by other DLS-user entities	see 7.2
Keeper object	accessed via the Keeper UCMM, or general UCMM fixed tag, to provide a range of support services for link operation, schedules and connections holds link data and attributes for all devices on a link keeps copies of all connection originator data for connection originating devices in a network negotiates with other Keeper objects on a link to agree on master Keeper and backup Keeper roles when master Keeper distributes link attributes to all nodes and keeps backup Keepers synchronized regularly issues TUI Lpacket to ensure all stations are synchronized to the current links configuration manages the network resource semaphore enabling orderly modification of link attribute data	see 7.3
Scheduling object	provides services to external link scheduling software in support of read/write for connection and schedule information from involved devices stores the current schedule details and connection password in the Keeper	see 7.4
TCP/IP Interface object	provides the mechanism to configure the TCP/IP interface of a device	see 7.5
Ethernet Link object	maintains link-specific counters and status information for a physical Ethernet ISO/IEC 8802-3 port	see 7.6
DeviceNet object	holds and distributes station management data classes and instances for Ph and DL layers, also collects diagnostic information for use by other DLS-user entities	see 7.7
Connection Configuration object	provides an interface to create, configure and control connections in a device	see 7.8

6 Specific DLPDU structure, encoding and procedures

6.1 Modeling language

6.1.1 State machine description

Some of the components of the Data Link Layer are specified using a state machine description language. The action statement is expressed in C++. Other components do not require a state machine description and are specified using only C++.

A state machine is described using the following syntax, where * indicates repetition of one or more of the preceding component, and {} indicates an optional component:

```

state-machine :=
    <header-statements>
    state*
state :=
    "state:" <state-name>
    transition*
transition :=
    "event:" <event-expression>
    {"condition:" <condition-expression>}
    "destination:" <state-name>
    {"action:" <action-statements>}
    
```

State-names are legal C++ identifiers, event-expressions and condition-expressions are C++ expressions, and header-statements and action-statements are lists of C++ statements.

Header-statements include

- #include and #define;
- variable and interface declarations;
- subroutine definitions.

An event-expression indicates the specified event when the value of the expression is TRUE. An event is required to trigger a transition. A condition-expression qualifies a transition. A qualified transition occurs when its event occurs if the condition is true at the time of the event. In general, event-expressions are FALSE upon entry into a state. There are a few cases where an event-expression may be true upon entry to a state. In these cases, the transition takes place immediately.

Action statements are executed when the transition is triggered.

All the event triggers and conditions of the current state are evaluated continuously and concurrently until a given transition is enabled. All expressions and statements execute instantaneously. The following model illustrates the precedence of the various parts of a transition:

```
if(the event has occurred && the condition is true)
{
  do the actions;
  go to the destination;
}
```

Implementations may use other code or syntax, however the implementation performance shall be equivalent to that specified except for internal execution timing and local variables

6.1.2 Use of DLL- prefix

The primitives and parameters described in IEC 61158-3-2 use the following prefixes:

- DL- for data link
- DLS- for DL-link service
- DLMS- for DL-management service.

Within this Type 2: Data Link Protocol Specification, the generic prefix DLL- is used as equivalent to the appropriate service description prefix.

6.1.3 Data types

The data types used in the data link variables, parameters, state machines and example code are specified in IEC 61158-5-2.

The specification of data types corresponds to the notation of IEC 61131-3. A list of the elementary data types, their corresponding description and some of their valid values is shown in Table 9. Additional detail is specified in IEC 61158-6-2.

Table 9 – Elementary data types

Keyword	Description	Range	
		Minimum	Maximum
BOOL	Boolean		
SINT	Short Integer	-128	127
INT	Integer	-32 768	32 767
DINT	Double Integer	-2 ³¹	2 ³¹ -1
LINT	Long Integer	-2 ⁶³	2 ⁶³ -1
USINT	Unsigned Short Integer	0	255
UINT	Unsigned Integer	0	65 535
UDINT	Unsigned Double Integer	0	2 ³² -1
ULINT	Unsigned Long Integer	0	2 ⁶⁴ -1
REAL	Floating point		
LREAL	Long float		
ITIME	Duration (short)		
TIME	Duration		
FTIME	Duration (high resolution)		
LTIME	Duration (long)		
DATE	Date only		
TIME_OF_DAY or TOD	Time of day		
DATE_AND_TIME or DT	Date and time of day		
STRING	character string (1 octet per character)		
STRING2	character string (2 octet per character)		
STRINGN	character string (N octets per character)		
SHORT_STRING	character string (1 octet per character, 1 octet length indicator)		
SWORD	bit string - 8 bits		
WORD	bit string - 16-bits		
DWORD	bit string - 32-bits		
LWORD	bit string - 64-bits		

6.2 DLS user services

6.2.1 General

The DLL user services provided to the DLS user have the following interfaces used for the specified purposes. The related service definitions and time sequence diagrams are given in IEC 61158-3-2.

6.2.2 Connected mode and connectionless mode transfer service

6.2.2.1 Service encoding

The request and confirmation services used to queue the sending of Lpackets by the DLL shall be of the following forms.

- Connection mode transfers use Generic tag request and confirmation.

```

DLL_xmit_generic_request(
    IDENTIFIER id,
    USINT Lpacket[],
    UINT Lpacket_size,
    PRIORITY priority,
    USINT gentag[3] );

DLL_xmit_generic_confirm(
    IDENTIFIER id,
    TXSTATUS status );

```

b) Connectionless mode transfers use Fixed tag request and confirmation.

```

DLL_xmit_fixed_request(
    IDENTIFIER id,
    USINT Lpacket[],
    UINT Lpacket_size,
    PRIORITY priority,
    USINT service,
    USINT destID );

DLL_xmit_fixed_confirm(
    IDENTIFIER id,
    TXSTATUS status );

```

c) Connection mode transfers use Generic tag indication.

```

DLL_recv_generic_indication(
    USINT Lpacket[],
    UINT Lpacket_size,
    USINT gentag[3] );

```

d) Connectionless mode transfers use Fixed tag indication.

```

DLL_recv_fixed_indication(
    USINT Lpacket[],
    UINT Lpacket_size,
    USINT service,
    USINT sourceID );

```

6.2.2.2 Procedures for request and confirmation

The `Lpacket` parameter shall contain the DLSDU as an ordered sequence of octets. The `Lpacket_size` parameter shall specify the number of octets contained in the `Lpacket` array. The `QoS priority` shall specify onto which internal queue the request shall be placed and shall be one of `LOW`, `HIGH` or `SCHEDULED`. Only `Lpackets` on the `SCHEDULED` queue may be sent during the scheduled portion of the NUT. The unscheduled portion of the NUT shall be used to transmit `Lpackets` from the `HIGH` and `LOW` `QoS priority` queues, and may be used to transmit `Lpackets` from the `SCHEDULED` queue. The queues shall be prioritized such that, during an unscheduled transmit opportunity, the `SCHEDULED` queue is emptied first, then the `HIGH` `QoS priority` queue is emptied second, and finally the `LOW` `QoS priority` queue is emptied last.

The `IDENTIFIER`, `id`, shall correlate a confirmation with a corresponding request. The `status` parameter shall return the status of the attempted transmit and shall be one of `OK`, `TXABORT` or `FLUSHED`.

The `DLL_xmit_fixed_request` service queues an `Lpacket` with a two octet fixed tag for transmit. The `service` parameter shall specify the first octet of the fixed tag `Lpacket`. The `destID` parameter shall specify the second octet, which shall determine the MAC ID of the destination node.

The remaining parameter, `gentag`, of the `DLL_xmit_generic_request` service shall specify on which generic tag to transmit the `Lpacket`.

NOTE 1 The use of one or more queues for different priorities is an internal implementation choice.

NOTE 2 Assembly of the related `Lpacket` structure and the procedure for its sending on the link are described in their respective clauses (see 5.3 and 5.4).

6.2.2.3 Procedures for indication

These indications shall signal the delivery of an `Lpacket` with a good FCS. Only fixed tag `Lpackets` which have had their service enabled by a successful call to `DLL_enable_fixed_request` shall cause a `DLL_recv_fixed_indication`. Only generic tag `Lpackets` which have had their generic tag enabled by a successful call to `DLL_enable_generic_request` shall cause a `DLL_recv_generic_indication`.

The array of octets, `Lpacket`, shall contain the DLSDU as an ordered sequence of octets. The integer, `Lpacket_size`, shall specify the number of octets contained in the `Lpacket` array.

6.2.3 Queue maintenance service

6.2.3.1 Service encoding

Once queued, an `Lpacket` shall remain in a transmit queue until it is transmitted or de-queued. The request and confirmation services used to de-queue an `Lpacket`, before it is transmitted, shall be of the form:

```
DLL_flush-requests-by-QoS_request( PRIORITY priority );
DLL_flush-requests-by-QoS_confirm( PRIORITY priority );
DLL_flush_single_request( PRIORITY priority, IDENTIFIER id );
```

6.2.3.2 Procedures for request and confirmation

`DLL_flush-requests-by-QoS_request` shall cancel all pending transmits at the specified QoS priority. Each pending transmit shall immediately terminate with a confirmation containing the `FLUSHED` status. `DLL_flush_single_request` shall cancel a specific `Lpacket` identified by the `id` used to transmit the `Lpacket`. This service need not have a corresponding confirmation since the confirmation of the queued `Lpacket` shall serve this purpose.

6.2.4 Tag filter service

6.2.4.1 Service encoding

By default, the Data Link Layer shall only process the moderator `Lpacket` (fixed tag 0x00), and all other `Lpackets` shall be discarded. Other protocol layers may enable or disable reception of specific `Lpackets` using the following services;

<code>DLL_enable_generic_request(</code> <code>IDENTIFIER id,</code> <code>USINT gentag[3]);</code>	<code>DLL_enable_generic_confirm(</code> <code>IDENTIFIER id,</code> <code>BOOL result);</code>
<code>DLL_disable_generic_request(</code> <code>IDENTIFIER id,</code> <code>USINT gentag[3]);</code>	<code>DLL_disable_generic_confirm(</code> <code>IDENTIFIER id,</code> <code>BOOL result);</code>
<code>DLL_enable_fixed_request(</code> <code>IDENTIFIER id,</code> <code>USINT service);</code>	<code>DLL_enable_fixed_confirm(</code> <code>IDENTIFIER id,</code> <code>BOOL result);</code>
<code>DLL_disable_fixed_request(</code> <code>IDENTIFIER id,</code> <code>USINT service);</code>	<code>DLL_disable_fixed_confirm(</code> <code>IDENTIFIER id,</code> <code>BOOL result);</code>

6.2.4.2 Procedures for request and confirmation

The `IDENTIFIER, id`, shall correlate a confirmation with a corresponding request. The parameter, `result`, shall return `TRUE` if the request was successful and `FALSE` if unsuccessful. The `gentag` parameter shall specify which generic tag to route to the higher layer that enabled the tag; likewise, the `service` parameter shall specify which fixed tag to route to the higher layer that enabled the tag.

NOTE The `DLL_enable_generic_request` service can return an unsuccessful result if the implementation of the Data Link Layer is unable to filter any more generic tags.

6.2.5 Link synchronization service

6.2.5.1 Service encoding

The indication that a new network update time (NUT) has begun shall be of the form:

```
DLL_tone_indication( USINT cycle );
```

6.2.5.2 Procedures for indication

The parameter `cycle` shall return the interval counter from the moderator DLPDU just received.

NOTE If a moderator DLPDU is expected but does not arrive, the local node simulates the reception of the moderator DLPDU based on an internal timer of the ACM.

6.2.6 Synchronized parameter change service

6.2.6.1 Service encoding

All nodes on a link shall maintain two copies of link parameters: current and pending. The current copy of link parameters shall be used for the on going operation of the Data Link Layer. The pending copy shall be maintained to allow a synchronized change of link parameters. The local DLL user services that read and write these link parameters shall be of the form:

<code>DLL_set_pending_request(</code> IDENTIFIER id, DLL_config_data params);	<code>DLL_set_pending_confirm(</code> IDENTIFIER id, BOOL result);
<code>DLL_set_current_request(</code> IDENTIFIER id, DLL_config_data params);	<code>DLL_set_current_confirm(</code> IDENTIFIER id, BOOL result);
<code>DLL_get_pending_request(</code> IDENTIFIER id);	<code>DLL_get_pending_confirm(</code> IDENTIFIER id, DLL_config_data params);
<code>DLL_get_current_request(</code> IDENTIFIER id);	<code>DLL_get_current_confirm(</code> IDENTIFIER id, DLL_config_data params);

The `DLL_config_data` shall contain the link parameters and be of the form:

```
class DLL_config_data
{
public:
    USINT myaddr;           // the MAC ID of this node
    UINT  NUT_length;       // the length of the NUT in 10 µs increments
    USINT smax;             // highest MAC ID allowed to transmit scheduled
    USINT umax;             // highest MAC ID allowed to transmit unscheduled
    USINT slotTime;        // time allowed for line turnaround in 1 µs increments
    USINT blanking;        // time to disable RX after DLPDU in 1,6 µs increments
    USINT gb_start;        // 10 µs intervals from start of guardband to tone
    USINT gb_center;       // 10 µs intervals from start of moderator to tone
    USINT modulus;         // modulus of the interval counter
    USINT gb_prestart;     // transmit cut-off, 10 µs intervals before tone
};
```

The local DLL user services which request and confirm the start of a `tMinus` countdown shall be of the form:

<code>DLL-tMinus-start-countdown-request(</code> IDENTIFIER id, USINT start_count);	<code>DLL-tMinus-start-countdown-confirm(</code> IDENTIFIER id, BOOL result);
--	--

NOTE The `tMinus` fixed tag service (see 6.9) is used by the master Keeper to request all stations on the link to initiate a changeover countdown. This is the local DLL user procedure to request and confirm local participation.

At the correct expiry of the `tMinus` countdown, each participating station passes a local indication to its `ControlNet` object requesting a change from current link parameters to pending link parameters.

```
DLL_tminus_zero_indication( );
```